NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A COMPARISON OF AIRCRAFT DEPOT INDUCTION PROCESSES: ASPA AND PDM

by
Michael D. Walls
September, 1996

Thesis Advisor:	Arnold H. Buss
Thesis Co-Advisor:	Donald Eaton

REPORT DOCUMENTATION PAGE			Form approved OMB No. 0704-188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information including suggestions for reducing this burden, to Washington Headquarters services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.			
1. AGENCY USE ONLY (*Leave Blank*)	**2. REPORT DATE** September, 1996.	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE **A COMPARISON OF AIRCRAFT DEPOT INDUCTION PROCESSES: ASPA AND PDM (U)**			**5. FUNDING NUMBERS**
6. AUTHOR(S) Walls, Michael D.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION/AVAILABILITY STATEMENT Distribution authorized to DOD Components only; Proprietary Information Involved; September 1996; Other requests for this document must be referred to Superintendent, Code 043, Naval Postgraduate School, Monterey, CA 93943-5000 via Defense Technical Information Center, 8725 John J. Kingman Rd., STE 0944, Ft. Belvoir, VA 22060-6218.			**12b. DISTRIBUTION CODE**

13. ABSTRACT (Maximum 200 words)

The purpose of this thesis is to compare two predominant decision processes for aircraft depot inductions. The first process, Aircraft Service Period Adjustment (ASPA), is currently applied to the majority of Naval aircraft. The decision to induct an aircraft through ASPA is based on the results of subjective, periodic inspection. The second process, Programmed Depot Maintenance (PDM), is used by the U.S. Air Force. The Navy is also experimenting with its own form of PDM, called Phased Depot Maintenance. The PDM concept is based on the idea that regular overhaul of aircraft reduces man-hour requirements, turn-around time, and the variability of planning factors. The decision to induct an aircraft under PDM is entirely objective, as it is based solely on calendar time. A statistical comparison of the long term effects of ASPA and PDM is achieved by analyzing the output data of a simulation model designed in this thesis. Model output includes maintenance man-hour and turn-around time per depot overhaul, and the cumulative time in the depot and number of depot periods over the length of the simulation. The analysis provides insight into the benefits and trade-offs involved with each decision process.

14. SUBJECT TERMS Aviation, Depot, ASPA, PDM, Simulation			**15. NUMBER OF PAGES** 86
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF THIS ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18

A COMPARISON OF AIRCRAFT DEPOT INDUCTION PROCESSES: ASPA AND PDM

Michael D. Walls
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1986

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1996

Author: ______________________________
Michael D. Walls

Approved by: ______________________________
Arnold H. Buss, Thesis Advisor

____
Donald Eaton, Thesis Co-Advisor

____
Samuel H. Parry, Second Reader

____
Frank Petho, Chairman, Department of Operations Research

ABSTRACT

The purpose of this thesis is to compare two predominant decision processes for aircraft depot inductions. The first process, Aircraft Service Period Adjustment (ASPA), is currently applied to the majority of Naval aircraft. The decision to induct an aircraft through ASPA is based on the results of subjective, periodic inspection. The second process, Programmed Depot Maintenance (PDM), is used by the U.S. Air Force. The Navy is also experimenting with its own form of PDM, called Phased Depot Maintenance. The PDM concept is based on the idea that regular overhaul of aircraft reduces man-hour requirements, turn-around time, and the variability of planning factors. The decision to induct an aircraft under PDM is entirely objective, as it is based solely on calendar time. A statistical comparison of the long term effects of ASPA and PDM is achieved by analyzing the output data of a simulation model designed in this thesis. Model output includes maintenance man-hour and turn-around time per depot overhaul, and the cumulative time in the depot and number of depot periods over the length of the simulation. The analysis provides insight into the benefits and trade-offs involved with each decision process.

Thesis Disclaimer

The reader is cautioned that assumptions made with regard to the data used in this research are those of the author. Furthermore, although every effort has been made to ensure that the computer simulation program is free of computational and logical errors, it cannot be considered validated. Any application of information obtained from this thesis without further validation is at the risk of the user.

TABLE OF CONTENTS

I. INTRODUCTION 1

A. BACKGROUND 1

1. History 1

2. EA-6B Prowler 2

B. OBJECTIVE 3

II. DEPOT INDUCTION PROCESSES 5

A. INTRODUCTION 5

B. NAVAL AVIATION MAINTENANCE 5

C. AIRCRAFT SERVICE PERIOD ADJUSTMENT (ASPA) 6

D. PROGRAMMED DEPOT MAINTENANCE (PDM) 9

E. COMPARISON, ASPA AND PDM 10

F. OTHER CONSIDERATIONS 11

1. Major Repair 12

2. Design Modifications 12

G. PHASED DEPOT MAINTENANCE 13

III. METHODOLOGY 15

A. INTRODUCTION 15

B. SIMULATION 15

IV. SIMULATION DESCRIPTION 17

A. ASSUMPTIONS 17

B. INPUT 17

1. User Input to the Simulation 17

2. File Input to the Simulation 17

C. MODELED OBJECTS 18

1. Operations 18

2. Aircraft 18

3. Screen 18

4. Inspector 19

5. Depot 19

6. Depot Manager 20

D. GENERATION OF FLIGHT DATA 20

E. GENERATION OF ASPA FAILURE RATES 20

F. GENERATION OF FATIGUE LIFE EXPENDED (FLE) 21

G. GENERATION OF MAINTENANCE MAN HOURS (MMHRS) 21

V. MODEL IMPLEMENTATION 23

A. BASELINE SIMULATION RUN 23

B. SIMULATION OUTPUT AND ANALYSIS 23

C. SENSITIVITY ANALYSIS 25

D. DISCUSSION 27

VI. CONCLUSIONS AND RECOMMENDATIONS 29

A. CONCLUSIONS AND RECOMMENDATIONS 29

B. FURTHER RESEARCH 30

LIST OF REFERENCES 31

APPENDIX A. 33

APPENDIX B. 35

APPENDIX C. 37

APPENDIX D. 39

APPENDIX E. 41

APPENDIX F. 43

INITIAL DISTRIBUTION LIST 85

EXECUTIVE SUMMARY

The majority of aircraft type model series in the current Navy inventory are expected to remain in operational service well beyond the original intent of design engineers. These aircraft require periodic depot level maintenance, or overhauls, to ensure that they are structurally sound and capable of meeting operational requirements.

The current process used for determining which Naval aircraft should be inducted into aviation depots for overhaul is the Aircraft Service Period Adjustment Program (ASPA). ASPA is outwardly logical in that the intent of the program is to prevent the unnecessary maintenance of aircraft. Although there are specific ASPA inspection guidelines, there is an inherent subjectivity to the process which has had a detrimental effect on operational readiness. Many aircraft remain in operational service for eight to ten years. When the aircraft finally arrive at the depot for basic overhaul, numerous unplanned maintenance problems, or "over and aboves," are discovered. These over and above issues require substantial amounts of additional maintenance hours and material to rectify. Increased maintenance and material requirements translate directly to reduced operational availability and increased cost.

An alternative to ASPA is a process by which the decision to induct aircraft into depots is based solely on calendar time in operation. This process, known as Programmed or Phased Depot Maintenance (PDM), removes the subjectivity of the ASPA inspection by strict adherence to an established induction schedule. The benefits of PDM are realized in lower and more predictable levels of maintenance hours and material. The resultant consistency in planning factors allows for more accurate planning by both operational and maintenance planners. The extended benefits of PDM are increased operational availability of aircraft and reduced cost for depot overhauls.

The objective of this study is a comparison of the effectiveness of ASPA and PDM. Effectiveness is expressed in terms of the maintenance man-hours and turn-around time required for a depot overhaul period. The basis for this study is the EA-6B Prowler, the premier tactical electronic warfare platform in the U.S. arsenal. A computer model was designed to simulate the long term effects of each policy on the EA-6B inventory. The model emulates fleet-to-depot cycles of each EA-6B over a 30 to 40 year period. A simulation run is conducted independently for each induction process under the same conditions. A statistical comparison of measures of effectiveness (MOE) is conducted to provide insight into the benefits and trade-offs that are realized from each process. The MOE's used in the comparison are average maintenance man-hour requirement and turn-around time per depot overhaul, and the cumulative depot times and number of depot periods per aircraft over the length of the simulation.

The results of the simulation show that a PDM overhaul requires fewer man-hours and reduces turn-around time per aircraft than does ASPA. The results also show that PDM planning factors are less variable and therefore more predictable. The model output indicates that the cumulative depot time and number of depot periods increase under PDM, since there is no chance for induction deferral. This result suggests a trade-off between a greater number of shorter, less intensive depot periods and a fewer number of longer, more intensive depot periods. The results of this model can provide decision makers with valuable insight into the effects of ASPA and PDM policies. This insight may be used when determining which process to apply to future type model series aircraft.

I. INTRODUCTION

A. BACKGROUND

1. History

Modern military aircraft are often in operational service well beyond their original design specifications. The Boeing B-52 Stratofortress flew for the first time in April, 1952 and will operate in the United States Air Force beyond the year 2000 [Ref 1, p. 107]. The Grumman A-6E Intruder entered service in the United States Navy in 1963 and will not be completely retired until early 1997 [Ref 2, p. 405]. As aircraft remain in service beyond their expected lifetimes, extensive overhaul maintenance becomes essential.

Overhaul maintenance is expensive, time consuming, and requires specialized equipment and facilities beyond the capabilities of operational units. The Department of Defense has used organic and commercial depots to overhaul and repair aircraft since 1941 [Ref 9, p.4]. At the height of the defense build-up of the 1980's, the Department of the Navy had an extensive depot network capable of serving the wide variety of aircraft and systems operating in the fleet. Aircraft assigned to the Pacific Fleet were serviced at Naval Aviation Depots Alameda and North Island located in California. Atlantic Fleet aircraft were serviced at NADEP Norfolk, Virginia, NADEP Jacksonville, Florida and NADEP Pensacola, Florida. The depot at MCAS Cherry Point, North Carolina also serviced Navy and Marine Corps aircraft.

The Navy Depot system, however, has not faired well since the Base Realignment and Closure committee began targeting bases and installations for closure in an effort to reduce so called “excess infrastructure.” The Navy must now rely on three organic Aviation Depots to perform depot level maintenance on aircraft: NADEPs Jacksonville, North Island and Cherry Point. Excess workload may in some cases be assumed by commercial sources like Grumman Facility at St. Augustine, Florida.

The services provided by the Navy depot system are critical to the operational readiness of fleet combat units. Maintaining the material condition of naval aircraft is an essential part of the readiness formula and the services provided by depots are critical to that purpose. At issue is the amount of time an aircraft spends in the depot over its life cycle and the cost of performing depot level repairs and overhauls. Ideally, operational units would always be able to turn in an aircraft to the depot and immediately receive a replacement. Budgetary realities make that prospect unlikely. Therefore, when an aircraft is inducted into a depot, the effects on readiness are immediate. Inventory managers are faced with the balancing aircraft operational availability and material condition while constrained by limited fiscal and material resources.

2. EA-6B Prowler

In 1995, the Air Force deferred responsibility for tactical electronic warfare to the Navy. Although the EA-6B Prowler has been in service since 1971, the expanded mission of the community will require the continued service of the aircraft until 2015. There are currently 127 EA-6B Prowler aircraft in the Navy Total Overall Aircraft Inventory (TOAI). Aircraft in the TOAI are distributed to operational units as a part of the primary aircraft inventory (PAI), or to the maintenance “pipeline” as a part of the backup aircraft inventory. By the year 2001, the Prowler TOAI will have decreased to 123 aircraft while

104 aircraft will be required to meet anticipated operational, training and testing commitments. To meet the expected demand and inevitable unexpected maintenance burdens of the next 20 years, inventory managers will have to maintain the highest levels of aircraft material condition while maintaining availability requirements.

Many of the EA-6B airframes currently in the fleet have been in operational status for five or more years. In some cases, aircraft have not been in the depot for ten years. The EA-6B inventory managers are now faced with the dilemma of inducting aircraft into the depot for desperately needed overhaul at the expense of aircraft availability to the fleet.

B. OBJECTIVE

There are currently two predominant aircraft depot maintenance philosophies ascribed to in the Department of Defense. The first philosophy suggests that the decision to induct an aircraft into an aviation depot for overhaul be based on subjective interpretation of general material condition guidelines. The second philosophy prescribes induction on a calendar schedule without regard to apparent material condition. The purpose of this project is to provide some insight into the effects of each process on labor requirements and aircraft down time. An understanding of these issues will assist the aircraft inventory manager in making sound readiness and fiscal decisions.

II. DEPOT INDUCTION PROCESSES

A. INTRODUCTION

The first section of this chapter is a brief description of the Naval Aviation maintenance concept. Subsequent sections describe the Aircraft Service Period Adjustment (ASPA), Programmed and Phased Depot Maintenance (PDM) and other decision considerations used in the aircraft depot induction process.

B. NAVAL AVIATION MAINTENANCE

The Naval Aviation maintenance concept is centered around a three tier system. Figure 1 depicts the flow of repairable parts and systems through the three levels of maintenance. Operational maintenance is performed by technicians assigned to squadron or wing level units and is usually limited to basic servicing of the aircraft and removal and replacement of faulty components. Components removed at this level are sent to an intermediate maintenance facility. Intermediate maintenance facilities are located at most Naval Air Stations and aboard aircraft carriers. Here components are repaired and held until issued to operational units when required. If the intermediate facility is incapable of repairing the component it may be forwarded to the final tier of the system, the depot. Aircraft components or systems are inducted into the depot for overhaul, extensive repair or engineering modification. Overhaul is referred to as Standard Depot Level Maintenance or SDLM. The specific tasks performed during a SDLM are based on extensive reliability centered maintenance and outlined in the type model series (TMS) SDLM Specification.

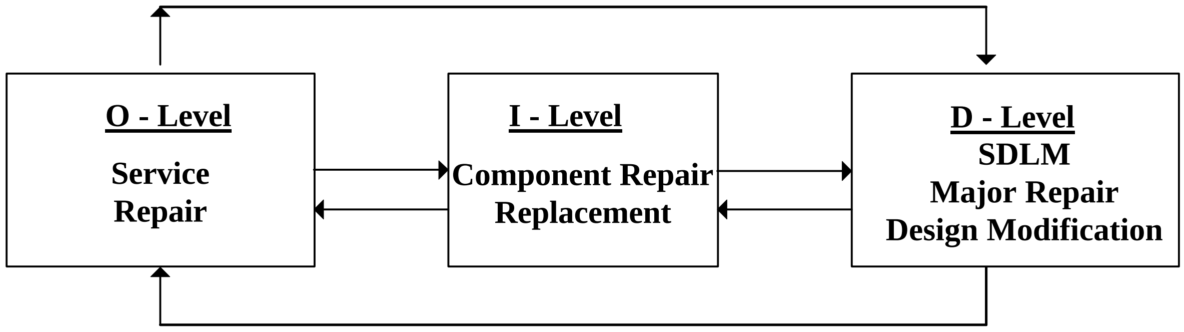


Figure 1. *Naval Aviation Levels of Maintenance*

C. AIRCRAFT SERVICE PERIOD ADJUSTMENT (ASPA)

The Department of the Navy instituted the Aircraft Service Period Adjustment, ASPA, program in 1982. The philosophy of ASPA was driven by a desire to avoid inducting aircraft of sound material condition into the depot for overhaul. The concept was supported by the idea that the material condition of individual aircraft do not deteriorate at the same rate. Aircraft should be inducted on the basis of “on condition” rather than “on time” [Ref 4, p. 2].

Each TMS in the Navy inventory is assigned an Operational Service Period, OSP, per OPNAVINST 3110.11T. The OSP defines the minimum time period between SDLM and “provides the basis for planning, programming and budgeting for the particular TMS” [Ref 4, p. 1]. A Planning and Estimator (P&E) team is dispatched from the Cognizant Field Activity (CFA) when an aircraft reaches its OSP. The objective of the P&E team is to asses the material condition of the aircraft and to determine suitability for continued service. The P&E team will forward an ASPA P&E Report to the aircraft controlling custodian, ACC, within three days of the inspection completion. The report will recommend immediate depot induction for SDLM, induction within 90 days of the current PED or adjustment of the PED to an additional 12 months. Subsequent ASPA inspections will be performed within PED minus 180 days and PED + 90 days. This cycle continues until the aircraft fails the material condition inspection. The induction of an aircraft can be further delayed if a waiver is issued by the Office of the Chief of Naval Operations (N881C) [Ref 5, p. 4].

The ASPA process is an outwardly logical method for determining if an aircraft should be inducted for overhaul. ASPA is supported by the idea that aircraft above a material condition threshold requiring SDLM should be kept in operational status. Resources should be preserved for those aircraft below that threshold. This is especially true in the Navy, where fiscal resources must be spread throughout all of the warfare and support communities. The data in Figure 2 depict ASPA inspection results for EA-6B aircraft in their first operational tour. The data show that of the 60 Prowlers inspected, most passed early ASPA inspections. By deferring induction, the aircraft were available for operations well beyond the original 54 month OSP. In many cases, aircraft were operational for 10 years.

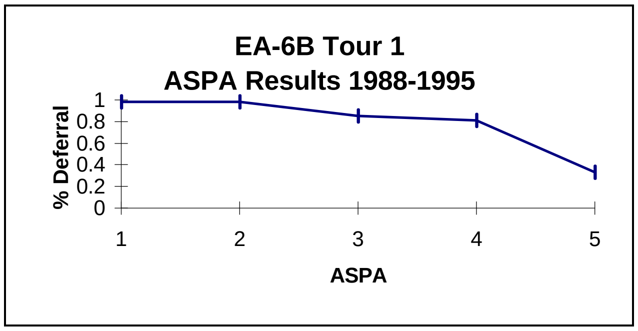


Figure 2. *EA-6B ASPA Inspection Results*

On the surface, ASPA provides an objective method for selecting aircraft for induction into the depot. ASPA induction recommendations are based on definitive criteria which are continually evolved to reflect current material and economic conditions [Ref 4, p 2]. There is, however, an inherently subjective nature to the process. The recommendations of the inspection team are based primarily on the sound judgment and expertise of the team members rather than rigid quantitative criteria. The final decision to induct is then based on that recommendation and the ACC's evaluation of the total readiness situation.

While the data in Figure 2 suggest that the Navy is preserving resources through ASPA, there are significant underlying implications. Maintenance man-hours (MMHRS) and turn-around time (TAT) are two key factors that are adversely affected by ASPA in the long run. Each of these factors becomes more unpredictable as the time between depot rework periods increases. Figure 3 shows the relationship between MMHRS and the time between depot reworks, IDT, for the EA-6B from 1980 to 1995. The data show that the variability of MMHRS increases significantly as IDT increases beyond the six year point. A similar plot for TAT is included in Appendix A.

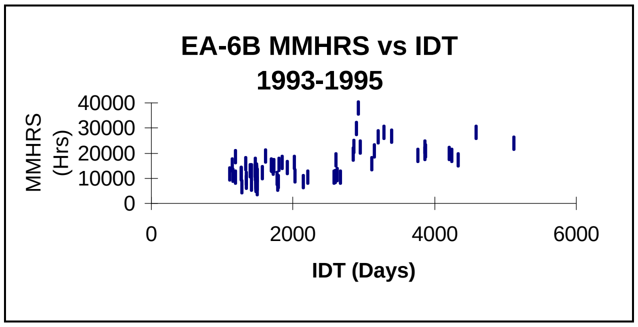


Figure 3. *MMHRS:* m= *23,496 hrs /* s *= 5,223 hrs*

Further inspection of the data in Figure 3 reveals an increasing trend in the number of MMHRS required for a rework period as the inter-depot time increases beyond approximately 2500 days. The same trends are apparent for TAT and cost. The behavior exhibited by the Prowler is not unique. Thirty-nine F-14 Tomcats were inducted for SDLM between 1991 and 1994. The mean man-hours per aircraft was 28,127 hrs with a standard deviation of 4,146 hrs [Ref 3, p.40]. The long run effects of deferring aircraft depot inductions are increased MMHRS and TAT which inevitably result in higher cost and decreased availability of operational aircraft.

D. PROGRAMMED DEPOT MAINTENANCE (PDM)

The Navy ASPA program is not practiced by either the United States Air Force or the Commercial Airline industry. These groups employ a system for inducting aircraft into depots for overhaul based on service time rather than the subjectivity of an inspection team. The Air Force version of PDM, in this case Programmed Depot Maintenance, requires that aircraft be inducted into the depot for overhaul after reaching an established flight hour threshold. The objective of PDM is to ensure that the most mission capable aircraft are produced through expeditious and economic depot processes [Ref 3, p. 5]. Like the Navy, PDM is used in an effort to minimize the effect of overhaul maintenance on combat readiness. PDM differs from ASPA most significantly by removing the subjectivity in the induction decision process. Each aircraft shall be inducted on the basis of calendar time or flight hour threshold. If an aircraft is not inducted within the designated time window, PDM date +/- 90 days, the aircraft is grounded. Figure 4 shows PDM data for the General Dynamics EF-111 Raven, the U.S. Air Force tactical electronic warfare aircraft. Each point on the graph represents one of 24 observations of man hour requirements for EF-111 overhauls. Each of the 24 aircraft were inducted into the depot at the 1,500 hour mark, roughly equivalent to four years.

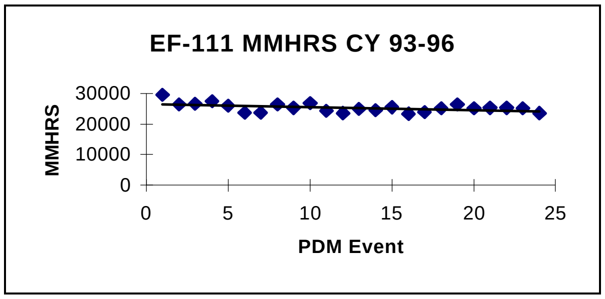


Figure 4. *MMHRS:* m= *25,311 hrs /* s *= 1438 hrs*

The data show that the man hour requirement for PDM's on most of the aircraft in the sample was relatively constant. The plot depicting TAT's for the sample aircraft, Appendix B, shows similar results. Reduced variability of these critical planning factors is an obvious advantage to aircraft logistics planners.

E. COMPARISON, ASPA AND PDM

The ASPA program is conceptually logical. Unfortunately, the degree to which subjectivity is introduced into the depot induction process creates monumental planning and programming problems for aircraft inventory managers. PDM removes that subjectivity, thereby decreasing the variability of the planning factors used to determine optimal resource allocation. The following discussion offers a comparative example of the effects of ASPA and PDM on two similar aircraft.

While the EA-6B and the EF-111 are dissimilar airframes flown in different environments, the two aircraft are close in age and perform the same types of missions. With the exception of aircraft carrier catapults and landing, both aircraft undergo similar stresses in flight. This similarity offers an excellent opportunity to compare the results of ASPA and PDM with respect to the planning and programming factors introduced in Section C of this chapter.

Comparison of the standard deviation of MMHRS to the mean MMHRS required for the overhaul of each aircraft, Figures 2 and 3, offers some degree of quantitative evidence that ASPA results in a higher level of variability than PDM. Another, and perhaps more meaningful, measure of variability is the coefficient of variation [Ref 6, p.212]. The coefficient of variation, C_x, gives the standard deviation as a proportion of the mean of a sample. As C_x increases, the variability of the sample increases. This difference in variation depicted by Equation 1 offers more evidence that PDM is less variable, and therefore a better process for logistics planning than ASPA.

$$C_{EF\text{-}111} = 0.0568 < 0.2223 = C_{EA\text{-}6B} \qquad \text{EQ (1)}$$

Comparison of the coefficients of variation with respect to TAT produced similar results. Appendix C provides a detailed computation of the coefficients of variation for each aircraft.

F. OTHER CONSIDERATIONS

Two significant maintenance issues that may result in an aircraft being inducted into the depot are major repair and design modifications. These types of actions are usually too intensive and time consuming to be performed at the operational or intermediate levels. Aircraft inventory managers must consider these issues along with periodic overhaul when developing planning and programming factors.

1. Major Repair

The most significant depot level repair issue facing the EA-6B community is replacement of the wing center section due to expended fatigue life. A material becomes fatigued as it undergoes inflight aerodynamic stresses. Fatigue Life Expended, FLE, is a measure of the percentage of service life that a material already used and represents the level of fatigue damage that the material has accumulated [Ref 8, p.7]. Fatigue is an insidious problem because it is visually undetectable until catastrophic failure of the material occurs. FLE is determined by mathematical models and serves as a predictor of impending structural failure. The wing center section of the EA-6B is especially vulnerable to fatigue because of the design of the wing and in some cases, the material of which the wing is made. Wing center sections on the Prowler are made from one of two materials. The materials do not expend their lifetimes at the same rate.

2. Design Modifications

In an effort to maintain the war fighting advantage through advanced technology, the EA-6B has undergone several design modifications or “Block Upgrades.” There are currently three different configurations of the EA-6B operating in the fleet. Another configuration is scheduled for introduction beginning in 1997. Block upgrades are usually incorporated at a depot, but may be by a depot field team.

G. PHASED DEPOT MAINTENANCE

By the late 1980's, most Naval aircraft had been in service well beyond their original design specifications. Since the introduction of ASPA, the material condition of aircraft began to decline and the amount organizational maintenance requirements began to detract from fleet-wide combat readiness. The Navy responded to the problem by searching for different ways of managing aircraft through their expanding life cycles. One answer was Phased Depot Maintenance, similar to the Air Force version of PDM. Navy PDM differs from the Air Force process in that the Navy induction depot criteria are based primarily upon calendar time rather than flight hours. The basic principles are, however, the same; overhaul aircraft on a scheduled basis to reduce the variability of planning and programming factors. Initial exploration for Navy PDM began in 1989 with an analysis of the P-3C Orion, a land based, multi-engine maritime patrol aircraft. It has been in service since 1970 and is expected to continue operations until 2015. The original service life of the P-3C has been extended from 30 to 50 years. The P-3C PDM process requires induction of aircraft at four year intervals, with a complete PDM cycle covering 12 years [Ref 3, p.29].

The P-3C PDM program is performing well. The planned TAT for P-3C PDM was 145 calendar days in CY 95 and 126 days in CY 96. The mean TAT's during each calendar year were within two months of the planned goals; μ_{95} = 183 days, σ_{95} = 22 days and μ_{96}= 148 days, σ_{96} = 35 days. Figure 5 [Ref 12] graphically depicts that actual TAT's are converging toward planned goals. The long run effects are clear; shorter times in the

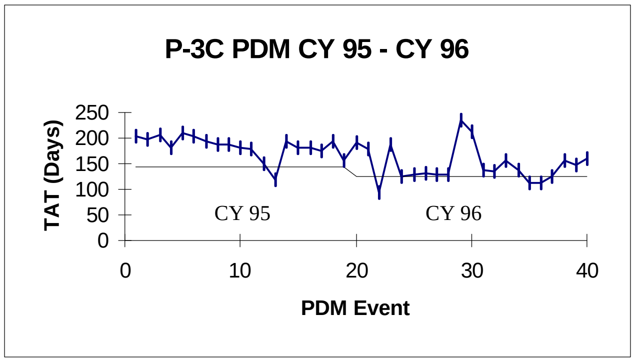


Figure 5. *P-3C PDM, Solid lines represent planned TATs*

depot for aircraft result in increased operational availability, and decreased variability in planning factors result in decreased costs.

III. METHODOLOGY

A. INTRODUCTION

This chapter describes the concept and development of the model. Later chapters will expand in detail on the formulation of the simulation.

B. SIMULATION

The goal of the simulation, written in MODSIM II [Ref 7], is to compare the long run effects of ASPA and PDM as processes for inducting EA-6B aircraft into aviation depots for overhaul. The simulation models the operational availability of a sample of EA-6B's over a 30 year lifetime while employing either ASPA or PDM. Operational availability is measured as the amount of time an aircraft is in operational status or the amount of time an aircraft is not in the depot. Simulation time is based on months.

The sample of aircraft was taken from the current inventory of operating Prowlers. The simulation commences with the delivery of the first lot of aircraft, with additional lots of aircraft incorporated at the beginning of subsequent months. The delivery dates of aircraft in the simulation correspond to the actual delivery month of the aircraft. The staggering of deliveries emulates the actual aircraft production and delivery process. Each aircraft enters the simulation with zero values for all flight data. Flight data for each aircraft are updated on a monthly basis.

At the end of a flying month, each aircraft is evaluated for suitability to continue operating. If the evaluation is negative, the aircraft is removed from operational status and inducted into an aviation depot. When depot space is available, the aircraft enters the maintenance process. The amount of time an aircraft spends in the maintenance process is based on the criteria on which it was inducted as is shown in Figure 6. When all maintenance

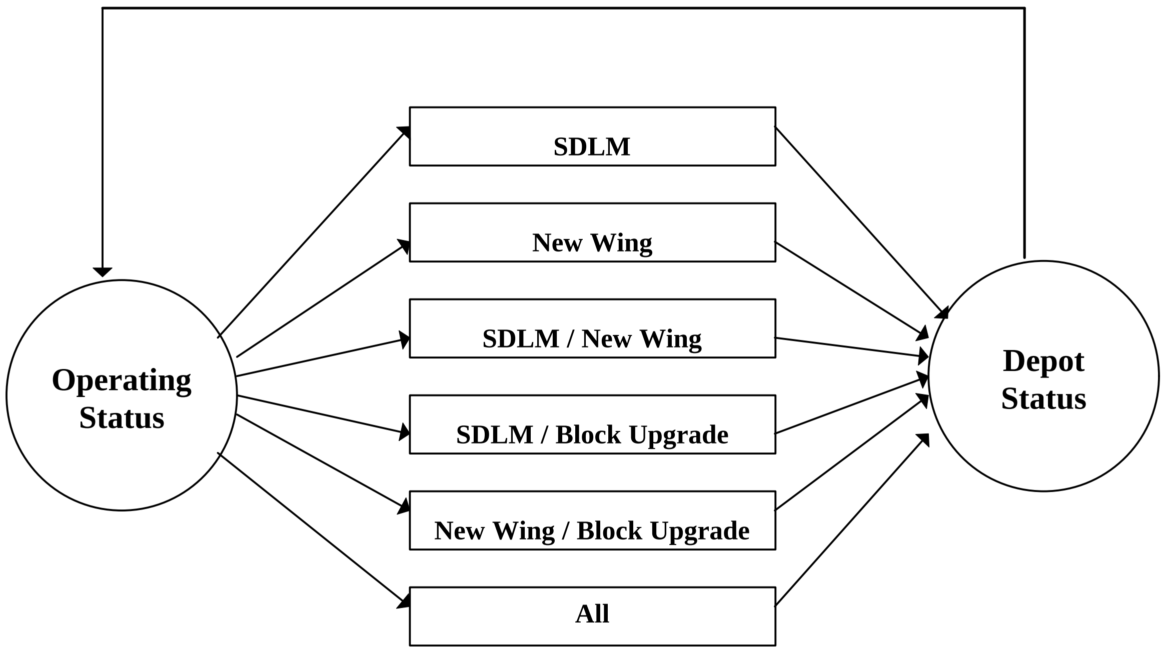


Figure 6. *Depot Induction Decision Criteria*

actions are complete, the aircraft returns to operational status and the depot reclaims the space resource. Turn-around time and maintenance man-hours are recorded after each depot period. The simulation is complete at the end of the final flying month.

IV. SIMULATION DESCRIPTION

A. ASSUMPTIONS

The following assumptions were made during development of the model:

1) Aircraft returning from the depot were considered to be new. Therefore, an aircraft had the same probability of failing a particular ASPA inspection regardless of the Tour.

2) Turn-around time is based on the maintenance man-months required for all maintenance activities and any queuing resulting from limited depot capacity.

B. INPUT

1. User Input to the Simulation

1) Simulation duration (months).

2) Desired depot induction process (ASPA or PDM).

3) Desired length of PDM cycle (months).

2. File Input to the Simulation

1) Aircraft groups (Lots) introduced at simulation times corresponding to actual service entry dates.

2) Aircraft Bureau (ID) number and type of wing (7050 or 7075).

3) Parameters for probability distributions used in random generation of monthly flight data.

4) Standard error used in the MMHRS regression model.

C. MODELED OBJECTS

The model was implemented in MODSIM II, a modular, object-oriented computer simulation language. This section describes the most significant objects developed for the simulation. Appendix F contains more detailed MODSIM II program code for the model.

1. Operations

One instance of the Operations Object, Operator, is called during the simulation to serve as the manager of the entire aircraft inventory. The Operator is responsible for invocation of the monthly Operations Method. During monthly operations, flight data are updated and depot screening is performed for operational aircraft. A depot status designation is assigned to aircraft that do not pass the screening process. The Operator is also responsible for opening the aviation depot.

2. Aircraft

Each Aircraft in the simulation is passively represented by an instance of the Aircraft Object. Randomly generated flight data are stored in designated fields of each aircraft. Each field is updated on a monthly basis. Various methods were programmed in the Aircraft Object for manipulation of data fields and use by other object entities in the simulation.

3. Screen

The Screen Object provides two methods for screening aircraft for depot induction. The ASPA-Screen Method makes depot induction determinations based on information obtained from emulated material condition inspections performed on each operational aircraft. The PDM-Screen Method evaluates each aircraft OSP. If an aircraft has reached the established OSP limit, it is inducted into the depot. Both screen methods assign job requirements which are stored in a record field of each aircraft object. Job requirements, listed in Figure 6 of Chapter III, are determined by evaluation of flight data, ASPA inspection results and service periods. Aircraft will not be inducted into the depot for the sole purpose of upgrading the configuration.

4. Inspector

The Inspector is responsible for performing ASPA inspections on each operational aircraft at the end of a flying month. Pass or Fail status is determined by comparing randomly generated uniform values and EA-6B ASPA failure rates. Results are stored in a data field of the aircraft being inspected. The Inspector is not used if the simulation is being run using PDM.

5. Depot

Aircraft are inducted into the Depot Object and processed according to the job requirements set during the depot screen. If the depot is full, aircraft wait in a queue until space becomes available. Processing times and maintenance man-hour requirements are based on actual EA-6B SDLM data from 1980 through 1995 [Ref 11]. Wing replacement and configuration upgrade processing times are also considered in the total processing time.

6. Depot Manager

The Depot Manager is responsible for determining annual depot capacity requirements. If aircraft are being inducted by the ASPA process, the Depot Manager records the total number of aircraft in each Tour/ASPA category at the beginning of each year. The number of spaces available is based on the expected percentage of inspection failures in each ASPA category. Failure rates are obtained from real EA-6B ASPA failure data currently used in the Depot Requirements Document (DERD) [Ref 10]. If PDM is the induction process being used on a simulation run, the Depot Manager bases depot capacity on the number of aircraft that are within 12 months of their OSP.

D. GENERATION OF FLIGHT DATA

Randomly generated values for various flight data are generated on a monthly basis. The values are based on the distributions assigned to each data field. The distributions were determined from analysis of six months of recorded flight data for every EA-6B in the Navy inventory. The descriptions and distributions of each flight data field are contained in Appendix E.

E. GENERATION OF ASPA FAILURE RATES

The ASPA failure rates are obtained from a Markov model developed from actual EA-6B ASPA inspection results from 1988 through 1995 [Ref 10]. The data contains deferral and non-deferral rates according to the Tour and ASPA number of the aircraft being inspected. The five states of the model are ASPA 1 through ASPA 5. The data are aggregated by ASPA number across all tours. The transition probability matrix given in

Appendix D is based on the individual state probabilities. To determine the outcome of an inspection, a random uniform number is generated and compared to an appropriate inspection failure rate. Pass or Fail status is conferred according to the results of the comparison.

F. GENERATION OF FATIGUE LIFE EXPENDED (FLE)

Each aircraft begins the simulation without any expended wing fatigue life. FLE is calculated monthly, and incrementally applied to each aircraft. The incremental value is based on the application of randomly generated flight data discussed in Section D and shown in Equation 2.

$$FLE = C_4\left(G_4 - G_5\right) + C_5\left(G_5 - G_6\right) + C_6\left(G_6 - G_7\right) + C_7\, G_7 + C_{Landings}\, Landings + C_{Cats}\, Cats \qquad \text{EQ (2)}$$

Values for the coefficients, C$_{\boldsymbol{i}}$, are based on the material type of the aircraft wing.
The G$_i$ values reflect the number of times a particular “G” level is exceeded during flight. Aircraft expending 100% of fatigue life will automatically be approved for a 600 flight hour extension.

G. GENERATION OF MAINTENANCE MAN-HOURS (MMHRS)

Values for MMHRS were generated from a log-linear regression model, Equation 3, developed by regressing the natural log of actual MMHRS [Ref 11] on the times between depot periods (IDT) for each aircraft. The log-linear design of the model captures the increase in variability of MMHRS as IDT increases. IDT, generated by the simulation, is measured from the time an aircraft leaves the depot after maintenance until it returns for subsequent maintenance.

$$ln\ MMHRS = \alpha + (\beta * IDT) + \varepsilon \qquad \text{EQ (3)}$$

A random error, ε, is generated from a normal distribution with mean 0 and variance equal to the fitted value from the log-linear regression. MMHRS are converted to maintenance man-months (MMTHS) by a factor based on the current expected maintenance man-hour and turn-around time requirements for an EA-6B overhaul at NADEP Jacksonville, Florida.

V. MODEL IMPLEMENTATION

A. BASELINE SIMULATION RUN

To compare the long run effects of implementing the ASPA and PDM processes for aircraft depot induction, the simulation was run separately for each process using the same input parameters given in Table 1.

Replications	150
Run Duration	30 Years
Number of Aircraft	109
PDM Cycle	54 Months

Table 1. *Baseline Input Parameters*

Because of the random nature of simulation output, the number of replications is set to ensure a minimum of 10% precision with 95% confidence. The 30 year run duration is an estimate of the lifetime of an average EA-6B. The first lot of aircraft was delivered at t = 0 and the final lot entered service at t = 253, or the 21 year mark. The initial PDM cycle of 54 months was chosen to corresponded to the current EA-6B OSP used in the ASPA process. Flight data distributions and MMHRS requirements generation were consistent for both processes.

B. SIMULATION OUTPUT AND ANALYSIS

An example of the simulation output is depicted in Table 2. The output reflects the mean value and standard deviation of each measure of effectiveness (MOE), per aircraft, over the length of a simulation run. Depot Time and Depot Periods represent the total time spent in the depot and the total number of depot periods, per aircraft, over the length of the simulation run. At the end of the baseline simulation run, there are 150 samples for each MOE.

Run	μ_{MMHRS}	σ_{MMHRS}	μ_{TAT}	σ_{TAT}	$\mu_{DepotTime}$	$\sigma_{DepotTime}$	$\mu_{DepotPeriods}$	$\sigma_{DepotPeriods}$
1	11029.70	3521.25	5.11	1.63	19.11	8.78	3.84	1.74
2	11368.76	3652.11	5.27	1.70	19.56	8.93	3.84	1.72
3	11334.47	3661.58	5.25	1.70	19.42	8.26	3.84	1.75
4	11043.21	3609.72	5.12	1.68	19.06	9.11	3.86	1.71
5	11426.57	3620.63	5.29	1.67	19.78	9.12	3.87	1.74
6	11428.34	4135.88	5.29	1.90	19.57	8.95	3.84	1.71
7	11294.50	3575.24	5.23	1.67	19.60	8.74	3.88	1.76
8	11241.59	3713.61	5.21	1.71	19.31	9.06	3.80	1.72
9	11348.21	3715.99	5.26	1.70	19.49	8.55	3.85	1.73
10	11188.28	3495.91	5.19	1.62	19.30	9.00	3.88	1.72

Table 2. *Sample Simulation Output, 10 of 150 Runs*

The estimates of the mean values for each MOE are depicted in Table 3. The last column represents the 95% Paired-t Confidence Interval for the difference between MOE’s of each process [Ref 13, p 586]. The confidence interval is an indication of the magnitude of that difference. A negative sign in front of an interval indicates that the ASPA value is less than the PDM value.

MOE	μ - ASPA	μ - PDM	95% CI
MMHRS	13561.41	11295.23	[1971.53 - 2560.84]
TAT (Mths)	6.3	5.2	[1.0 - 1.2]
Depot Time (Mths)	16.0	19.5	- [3.4 - 3.6]
Depot Periods	2.6	3.9	- [1.2 - 1.3]

Table 3. *MOEs and Paired-t Confidence Interval test.*

Review of the simulation output clearly indicates that over the length of the simulation, the MMHRS and TAT requirements for PDM aircraft are less than those of ASPA aircraft. Additionally, a comparison of coefficients of variation indicate that these planning factors are more predictable for PDM than for ASPA. Equation 4 is a comparison of the variability of MMHRS required for ASPA and PDM.

$$C_{PDM} = 0.3272 < 0.4072 = C_{ASPA} \qquad \text{EQ (4)}$$

The output also indicates that the cumulative time an aircraft will spend in the depot will be slightly greater for PDM than for ASPA. This result is intuitive because in the long run, adherence to a PDM induction schedule will result in an increased number of depot periods and a subsequent increase in total depot time.

C. SENSITIVITY ANALYSIS

Additional simulation runs were performed to test the sensitivity of the model to changes in the values of significant input parameters. In addition to testing the feasibility of the model, sensitivity runs also provide amplifying information about the behavior of the MOE's as conditions change. For this experiment, a 2^k factorial design was used to explore the effects of varying four parameters: simulation duration, standard error of the

MMHRS regression model (Equation 3), the PDM cycle and ASPA failure rates. Table 4 shows the range of values of the input parameters used in the sensitivity analysis.

	Duration (Mths)	MMHRS Std Err	PDM Cycle (Mths)	Failure Rate
High	480	0.6078	72	0.2
Low	240	0.0078	36	-0.2

Table 4. *Sensitivity Analysis - Input Parameters*

Results of the sensitivity analysis, Table 5, reveal that the model is extremely sensitive to changes in the standard error of the MMHRS regression model. As the error is increased, the MMHRS for both ASPA and PDM change significantly. Values for TAT and cumulative depot time are similarly affected.

The analysis also shows that PDM results are significantly affected by the length of the operational cycle (time between depot periods). The impact of IDT on the performance measures provides valuable insight into the desirable lengths of aircraft operational tours. Extended IDT's will increase both labor requirements and the length of depot periods. However, the magnitude of the PDM coefficients for both MMHRS and TAT suggests that there is a potential trade-off between aircraft availability to the fleet and a willingness to commit more maintenance resources to periodic overhauls. The increase in MMHRS incurred by a moderate increase of the PDM cycle length may be acceptable.

The effect of varying ASPA failure rates is also of interest. Although statistically insignificant, the negative values of the coefficients suggest that an increase in the number of ASPA failures results in a decrease in MMHRS, TAT and total depot time. This is precisely the result expected from a purely objective ASPA program. If ASPA is working properly, an aircraft will be inducted into the depot only when its condition warrants failure, resulting in fewer MMHRS and shorter depot overhaul periods. However, the results of the baseline simulation run, supported by the sensitivity analysis, suggest that there is a certain degree of subjectivity and inconsistency inherent in the ASPA program.

	MMHRS		**TAT**		**Depot Time**		**Depot Periods**	
	ASPA	**PDM**	**ASPA**	**PDM**	**ASPA**	**PDM**	**ASPA**	**PDM**
Intercept	16431.29	7926.99	7.90	3.80	-17.03	-6.62	-2.93	0.2136
Duration	-9.1675	-2.88	-0.0049	-0.0018	0.0909	0.1137	0.0159	0.0224
MMHRS ε	3306.20	2917.34	1.4967	1.32	3.49	4.71	-0.0442	-0.1243
PDM Cycle	0.0000*	77.82	0.0000*	0.0372	0.0000*	-0.2473	0.0000*	-0.0717
Failure Rate	-101.73*	0.0000*	-0.1002*	0.0000*	-0.1023*	0.0000*	0.0025*	0.0000

Table 5. *Sensitivity Analysis (* Not statistically significant)*

D. DISCUSSION

The baseline simulation output and results of the sensitivity analysis provide a sound basis for the comparison of ASPA and PDM. The model output clearly shows the advantages of PDM. Consistent depot inductions at smaller intervals result in fewer MMHRS, shorter TAT and less variability of depot planning factors. However, the model also suggests that to achieve the benefits of PDM, the cycle length must be chosen carefully. If the cycle is too long, the results may be no better than ASPA. Consideration must also be given to the increased cumulative depot time and the number of depot periods that result from PDM. The model output reveals that consistent and manageable

MMHRS and TAT levels require more visits to the depot and a slight increase in the cumulative time an aircraft spends in the depot.

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS AND RECOMMENDATIONS

The results discussed in Chapter V clearly indicate that a phased or programmed depot maintenance process (PDM) is preferable to ASPA. Analysis of the model output reveals that if an aircraft is inducted into the depot on a consistent and reasonable PDM schedule, instead of by a subjective and inconsistent inspection procedure, maintenance man-hour (MMHRS) requirements and turn-around time (TAT) will be lower. The beneficial effects of reduced MMHRS and TAT include lower labor cost per aircraft overhaul and increased aircraft availability to the fleet. Additionally, predictable PDM planning factors will make aircraft inventory and depot management more tractable. A more consistent depot process, resulting from a properly implemented PDM program, will also reduce unplanned costs, or “over and aboves”.

The model output also shows that an appropriate depot cycle is imperative if PDM is to be successful. If the PDM cycle is too long, MMHRS and TAT requirements will not show improvement when compared to ASPA. If the PDM cycle is too short, the number of depot periods per aircraft may increase excessively, resulting in reduced operational availability.

Analysis of the model output presents strong evidence that, since its inception, the inherent subjectivity of the ASPA program has resulted in years of unnecessarily high depot labor requirements and decreased aircraft availability. The analysis offers further evidence that PDM is a viable alternative to ASPA. It is recommended that consideration be given to implementation of phased or programmed depot maintenance when defining the maintenance concept for future Naval aircraft.

B. FURTHER RESEARCH

This model is capable of providing insight into the relationship between the time between depot periods and MMHRS required for a depot overhaul. However, the model as currently implemented does not quantitatively address the issue of unplanned costs (over and aboves). Research on unplanned requirements and their ensuing effects on MMHRS and TAT is recommended. The modular design of the simulation model in this thesis will simplify incorporation of enhanced features developed by further research.

A second enhancement to the current model would be the explicit representation of costs. In its present form, the model does not directly consider cost, although it does provide insight into the fiscal implications of PDM and ASPA. Studies related to the monetary cost of PDM are recommended to provide additional data and information to decision makers and to further validate this model.

LIST OF REFERENCES

1. Bowman, Martin W., *Encyclopedia of U.S. Military Airpower*, Arms and Armour Press, 1980.

2. Polmar, W., *The Ships and Aircraft of the U.S. Fleet*, 14th ed., Naval Institute Press, 1987.

3. Ramsey, Robert G. and Legidakes, Leo J., *An Analysis of the Impact on Organizational and Depot Level Maintenance,* Master's Thesis, Naval Postgraduate School, Monterey, CA, December 1994.

4. Department of the Navy Instruction (OPNAVINST 4730.10A), *Aircraft Service Period Adjustment*, October 1991.

5. Department of the Navy Instruction (OPNAVINST 3110.11T), *Policies and Peacetime Planning Factors Governing the Use of Naval Aircraft*, February 1993.

6. Rice, John A., *Mathematical Statistics and Data Analysis,* 2nd ed., Duxbury Press, 1995.

7. MODSIM II The Language for Object-Oriented Programming, CACI Products Company, January 1993.

8. Naval Air Systems (Air 4.3.3.2), *EA-6B Aircraft Structural Appraisal of Fatigue Effects (SAFE) Program Quartely Report*, July 1995.

9. Locklear, G.D., *Issues of Depot Maintenance: Changes for the Future, Research Report*, Industrial College of the Armed Forces, Washington, DC, April 1994.

10. Naval Aviation Depot Operations Center, ASPA Report - EA-6B Inspections, FY88-95.

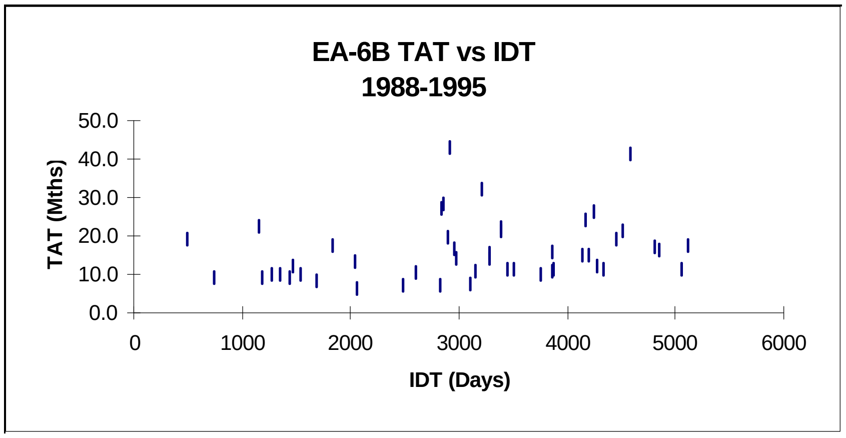
11. Naval Aviation Depot Operations Center, NADEP Production Performance Report Database, May 1996.

12. Naval Aviation Depot, Jacksonville, FL, P-3C Cost and Turn Around Time (TAT) Variance Report, June 1996.

13. Law, Averill M. and Kelton, David W., *Simulation Modeling and Analysis,* McGraw-Hill Inc., 1991.

APPENDIX A.

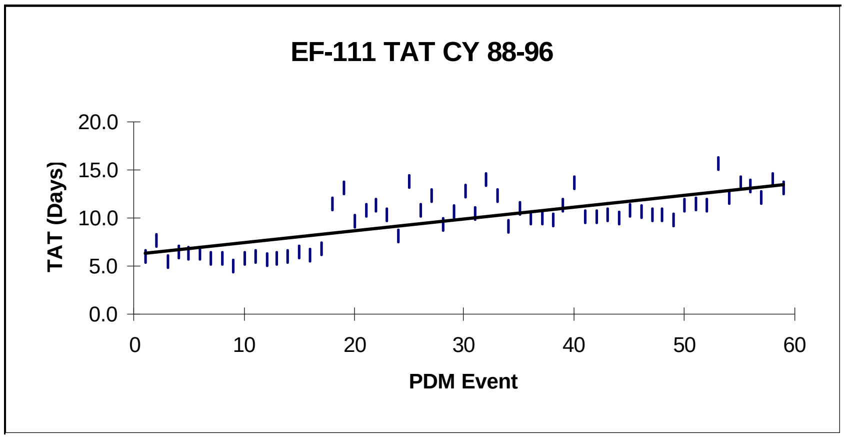
The following graph represents the relationship between turn-around time (TAT) and inter-depot time (IDT) for the EA-6B Prowler for the years 1988 through 1995.



TAT: μ = 16.0 Mths / σ = 8.3 Mths

APPENDIX B.

The following graph represents the variability of depot turn-around times (TAT) for the EF-111 Raven for the years 1988 through 1996.



TAT: μ = 9.9 Mths / σ = 2.8 Mths

APPENDIX C.

The following is a detailed computation of the coefficients of variation for maintenance man-hours (MMHRS) and turn-around time (TAT) for both the EA-6B and the EF-111.

MMHRS

$\mu_{\textbf{EA-6B}}$ = 23,496.28 Hrs

$\mathbf{C}_{\textbf{EA-6B}} = \sigma_{\text{EA-6B}} / \mu_{\text{EA-6B}} = 0.2223$

$\sigma_{\textbf{EA-6B}}$ = 5,223.54 Hrs

$\mu_{\textbf{EF-111}}$ = 25,311.39 Hrs

$\mathbf{C}_{\textbf{EA-6B}} = \sigma_{\text{EF-111}} / \mu_{\text{EF-111}} = 0.0568$

$\sigma_{\textbf{EF-111}}$ = 1,438.86 Hrs

TAT

$\mu_{\textbf{EA-6B}}$ = 17.2 Mths

$\mathbf{C}_{\textbf{EA-6B}} = \sigma_{\text{EA-6B}} / \mu_{\text{EA-6B}} = 0.5842$

$\sigma_{\textbf{EA-6B}}$ = 10.1 Mths

$\mu_{\textbf{EF-111}}$ = 9.9 Mths

$\mathbf{C}_{\textbf{EA-6B}} = \sigma_{\text{EF-111}} / \mu_{\text{EF-111}} = 0.2812$

$\sigma_{\textbf{EF-111}}$ = 2.8 Mths

APPENDIX D.

The following is a Markov model developed to provide probabilities of passing ASPA inspections. The data were obtained from actual EA-6B ASPA inspection results from 1988 through 1995 [Ref 10]. ASPA 5 inspections and greater will result in failure.

p_i = Probability that an aircraft will pass an ASPA inspection
i = ASPA State (1...4)

$p_1 = 0.9695$
$p_2 = 0.8952$
$p_3 = 0.8551$
$p_4 = 0.8298$

ASPA State Transition Matrix

ASPA	**1**	**2**	**3**	**4**	**5**
1	$1-p_1$	p_1	0	0	0
2	$1-p_2$	0	$-p_2$	0	0
3	$1-p_3$	0	0	p_3	0
4	$1-p_4$	0	0	0	p_4
5	1	0	0	0	0

APPENDIX E.

The flight data used in the simulation includes monthly statistics for each EA-6B in the Navy inventory for the period of January through June, 1995.

<u>Flight Data Distributions and Parameters</u>

Flight Hours ~	Normal (34.5,19.5)
Landings ~	Weibull (40.7,1.14)
Catapults ~	Weibull (2.63,0.62)
Four g ~	Gamma (24.6,79.9)
Five g ~	Beta (0.24,3.79)
***Six g ~**	Std Uniform / p = .1276
***Seven g ~**	Std Uniform / p = .0469

* If Std Uniform is less than p, a “g” value of 1.0 is scored.

APPENDIX F.

This Appendix includes the MODSIM II computer code referenced in Section C of Chapter V.

```
MAIN MODULE Prowler;

FROM Operations IMPORT OperationsObj;
FROM SimMod IMPORT StartSimulation, ResetSimTime, SimTime;
FROM Stats IMPORT StatsObj, CumStatsObj, YearlyStatsObj;

VAR
  RunNumber, PDMCycle, NumberOfRuns, Process  :  INTEGER;
  RunLength  :  REAL;
  FileName  :  STRING;
  Operator  :  OperationsObj;
  CumStatRecord  :  CumStatsObj;
  StatRecord:  StatsObj;
  YearlyStatRecord  :  YearlyStatsObj;
BEGIN
  OUTPUT("How many reps do you wish to run?");
  INPUT(NumberOfRuns);
  OUTPUT("How many months do you wish each run of the simulation to last?");
  INPUT(RunLength);
  OUTPUT("Simulate ASPA or PDM (1 = ASPA, 2 = PDM)");
  INPUT(Process);
  OUTPUT("Enter the length of the PDM cycle in months (0 if running ASPA)");
  INPUT(PDMCycle);
  OUTPUT("Enter the name of the output file.");
  INPUT(FileName);
  RunNumber := 1;
  NEW(Operator);
  ASK Operator TO PrepareReports(Process,FileName);
  WHILE NumberOfRuns > 0
    NEW(StatRecord);
    NEW(CumStatRecord);
    NEW(YearlyStatRecord);
    ResetSimTime(0.0);
    TELL Operator TO MonthlyOps(FileName,RunNumber,PDMCycle,RunLength,
                        Process,StatRecord,CumStatRecord,YearlyStatRecord);
    StartSimulation;
    DEC(NumberOfRuns);
    INC(RunNumber);
    DISPOSE(StatRecord);
    DISPOSE(CumStatRecord);
    DISPOSE(YearlyStatRecord);
  END WHILE;
  DISPOSE(Operator);
END MODULE.
```

```
DEFINITION MODULE Aircraft;

FROM Depot IMPORT DepotObj;
FROM NumberMill IMPORT FltDataGeneratorObj;
FROM Stats IMPORT StatsObj, YearlyStatsObj;

TYPE
  JobOrderType = (Rework,Rewing,ReworkRewing,ReworkUpgrade,
                         (RewingUpgrade,ReworkRewingUpgrade);

JobListRecType = RECORD
    TaskName  :  STRING;
    ExecuteTask,
    TaskRecord  :  BOOLEAN;
  END RECORD;

  JobListArrayType  =  ARRAY JobOrderType OF JobListRecType;

  DepotStatsType = RECORD
    NumberOfVisits,
    TimeInDepot,LaborMonths,LaborHours,TaskCost,
    CumTimeInDepot,CumLaborMonths,CumLaborHours,CumCost,
    IDT  :  REAL;
  END RECORD;

AircraftObj = OBJECT
  Status  :  STRING;
  Buno,
  Lndgs,CumLndgs,
  Cats,CumCats,
  FourG,CumFourG,
  FiveG,CumFiveG,
  SixG,CumSixG,
  SevenG,CumSevenG,
  WingType,
  Tour, ASPA,
  OSP,
  PED,
  Block,
  PDMCycle :  INTEGER;
  AircraftAge,Hrs,CumHrs,FLE,ExtHrs,UpdateFactor,DepotExitTime  :  REAL;
  OSPLimit,PEDLimit,FLELimit,EXTLimit,NeedBlock  :  BOOLEAN;
  DepotStats  :  DepotStatsType;
  JobList : JobListArrayType;
  ASK METHOD ObjInit;
  ASK METHOD CheckStatus() : STRING;
  ASK METHOD ChangeStatus(IN NewStatus : STRING);
  ASK METHOD GetAircraftAge() : REAL;
  ASK METHOD GetASPA() : INTEGER;
  ASK METHOD GetBlock() : INTEGER;
  ASK METHOD GetBuno() : INTEGER;
  ASK METHOD GetCumHrs() : REAL;
  ASK METHOD GetCumTimeInDepot() : REAL;
  ASK METHOD GetCumLaborMonths() : REAL;
  ASK METHOD GetCumLaborHours() : REAL;
  ASK METHOD GetExecuteTask(IN JobNumber : JobOrderType) : BOOLEAN;
  ASK METHOD GetExtHrs() : REAL;
  ASK METHOD GetEXTLimit() : BOOLEAN;
  ASK METHOD GetFLE() : REAL;
  ASK METHOD GetFLELimit() : BOOLEAN;
  ASK METHOD GetIDT() : REAL;
  ASK METHOD GetLaborMonths() : REAL;
  ASK METHOD GetLaborHours() : REAL;
  ASK METHOD GetNeedBlock() : BOOLEAN;
  ASK METHOD GetNumVisits() : REAL;
  ASK METHOD GetOSP() : INTEGER;
  ASK METHOD GetOSPLimit() : BOOLEAN;
  ASK METHOD GetPED() : INTEGER;
  ASK METHOD GetPEDLimit() : BOOLEAN;
  ASK METHOD GetTaskCost() : REAL;
  ASK METHOD GetTaskRecord(IN JobNumber : JobOrderType) : STRING;
  ASK METHOD GetTaskName(IN JobNumber : JobOrderType) : STRING;
  ASK METHOD GetTimeInDepot() : REAL;
  ASK METHOD GetTour() : INTEGER;
  ASK METHOD GetWingType() : INTEGER;
  ASK METHOD InitialData(IN DataString  :  STRING; IN Process : INTEGER);
  ASK METHOD InitialDepotFactors;
  ASK METHOD InitialDepotStats;
  ASK METHOD InitialPDMCycle(IN NumberOfMonths : INTEGER);
  ASK METHOD IncrementASPA;
  ASK METHOD IncrementTour;
  ASK METHOD ReInitializeASPA;
  ASK METHOD ReInitializePDM;
  ASK METHOD SetDepotStats(IN TAT, MMHRS, Cost : REAL);
  ASK METHOD SetExecuteTask(IN Condtion : BOOLEAN; IN JobNumber : JobOrderType);
  ASK METHOD SetPED;
  ASK METHOD SetTaskRecord(IN RecordCondition : BOOLEAN; IN JobNumber : JobOrderType);
  ASK METHOD SetTATs(IN TAT : REAL);
  ASK METHOD SetUpdateFactor;
  ASK METHOD UpdateAircraftAge;
  ASK METHOD UpdateCumData;
  ASK METHOD UpdateData(IN Gen  :  FltDataGeneratorObj);
  ASK METHOD UpdateDepotFactors;
  ASK METHOD UpdateDepotExitTime(IN ExitTime : REAL);
  ASK METHOD UpdateDepotStats;
  ASK METHOD UpdateFLE;
  ASK METHOD UpdateIDT(IN EntryTime : REAL);
  ASK METHOD UpdatePeriods;
  ASK METHOD UpdateStats(IN StatRecord : StatsObj; IN YearlyStatRecord : YearlyStatsObj);
  TELL METHOD GoToDepot(IN RunNumber, InductionProcess : INTEGER;
                        IN Generator : FltDataGeneratorObj;
                        IN Depot : DepotObj; IN StatRecord : StatsObj;
                        IN YearlyStatRecord : YearlyStatsObj);
END OBJECT;

END MODULE.
```

```
IMPLEMENTATION MODULE Aircraft;

FROM DataFinder IMPORT DataFinderObj;
FROM DataManager IMPORT DataManagerObj;
FROM Depot IMPORT DepotObj;
FROM IOMod IMPORT StreamObj, ReadKey;
FROM NumberMill IMPORT FltDataGeneratorObj;
FROM SimMod IMPORT SimTime;
FROM Stats IMPORT StatsObj, YearlyStatsObj;

OBJECT AircraftObj;
  ASK METHOD ObjInit;
    VAR
      DataFinder  :  StreamObj;
      ProcessTime  :  REAL;
      JobNumber  :  JobOrderType;
    BEGIN
      NEW(DataFinder);
      NEW(JobList,MIN(JobOrderType)..MAX(JobOrderType));
      FOR JobNumber := MIN(JobOrderType) TO MAX(JobOrderType)
        NEW(JobList[JobNumber]);
        CASE ORD(JobNumber)
          WHEN 0:
            JobList[JobNumber].TaskName := "Rework";
          WHEN 1:
            JobList[JobNumber].TaskName := "Rewing";
          WHEN 2:
            JobList[JobNumber].TaskName := "ReworkRewing";
          WHEN 3:
            JobList[JobNumber].TaskName := "ReworkUpgrade";
          WHEN 4:
            JobList[JobNumber].TaskName := "RewingRewing";
          WHEN 5:
            JobList[JobNumber].TaskName := "ReworkRewingUpgrade";
          OTHERWISE
            JobList[JobNumber].TaskName := "NA";
        END CASE;
        JobList[JobNumber].ExecuteTask := FALSE;
        JobList[JobNumber].TaskRecord := FALSE;
      END FOR;
    END METHOD;

  ASK METHOD CheckStatus() : STRING;
    BEGIN
      RETURN Status;
    END METHOD;

  ASK METHOD ChangeStatus(IN NewStatus : STRING);
    BEGIN
      Status := NewStatus;
    END METHOD;
```

```
 ASK METHOD GetAircraftAge() : REAL;
  BEGIN
    RETURN AircraftAge/12.0;
  END METHOD;

ASK METHOD GetASPA() : INTEGER;
  BEGIN
    RETURN ASPA;
  END METHOD;

ASK METHOD GetBlock() : INTEGER;
  BEGIN
    RETURN Block;
  END METHOD;

ASK METHOD GetBuno() : INTEGER;
  BEGIN
    RETURN Buno;
  END METHOD;

ASK METHOD GetCumHrs() : REAL;
  BEGIN
    RETURN CumHrs;
  END METHOD;

ASK METHOD GetCumTimeInDepot() : REAL;
  BEGIN
    RETURN DepotStats.CumTimeInDepot;
  END METHOD;

ASK METHOD GetCumLaborMonths() : REAL;
  BEGIN
    RETURN DepotStats.CumLaborMonths;
  END METHOD;

ASK METHOD GetCumLaborHours() : REAL;
  BEGIN
    RETURN DepotStats.CumLaborHours;
  END METHOD;

ASK METHOD GetExecuteTask(IN JobNumber : JobOrderType) : BOOLEAN;
  BEGIN
    RETURN JobList[JobNumber].ExecuteTask;
  END METHOD;

ASK METHOD GetExtHrs() : REAL;
  BEGIN
    RETURN ExtHrs;
  END METHOD;

ASK METHOD GetEXTLimit() : BOOLEAN;
  BEGIN
    RETURN EXTLimit;
  END METHOD;
```

```
ASK METHOD GetFLE() : REAL;
   BEGIN
     RETURN FLE;
   END METHOD;

 ASK METHOD GetFLELimit() : BOOLEAN;
   BEGIN
     RETURN FLELimit;
   END METHOD;

 ASK METHOD GetIDT() : REAL;
   BEGIN
     RETURN DepotStats.IDT;
   END METHOD;

 ASK METHOD GetLaborMonths() : REAL;
   BEGIN
     RETURN DepotStats.LaborMonths;
   END METHOD;

 ASK METHOD GetLaborHours() : REAL;
   BEGIN
     RETURN DepotStats.LaborHours;
   END METHOD;

 ASK METHOD GetNeedBlock() : BOOLEAN;
   BEGIN
     RETURN NeedBlock;
   END METHOD;

 ASK METHOD GetNumVisits() : REAL;
   BEGIN
     RETURN DepotStats.NumberOfVisits;
   END METHOD;

 ASK METHOD GetOSP() : INTEGER;
   BEGIN
     RETURN OSP;
   END METHOD;

 ASK METHOD GetOSPLimit() : BOOLEAN;
   BEGIN
     RETURN OSPLimit;
   END METHOD;

 ASK METHOD GetPED() : INTEGER;
   BEGIN
     RETURN PED;
   END METHOD;

 ASK METHOD GetPEDLimit() : BOOLEAN;
   BEGIN
     RETURN PEDLimit;
   END METHOD;
```

```
ASK METHOD GetTaskCost() : REAL;
  BEGIN
    RETURN DepotStats.TaskCost;
  END METHOD;

 ASK METHOD GetTaskName(IN JobNumber : JobOrderType) : STRING;
  BEGIN
    RETURN JobList[JobNumber].TaskName;
  END METHOD;

 ASK METHOD GetTaskRecord(IN JobNumber : JobOrderType) : STRING;
  BEGIN
    IF JobList[JobNumber].TaskRecord
      RETURN JobList[JobNumber].TaskName;
    ELSE
      RETURN "NA";
    END IF;
  END METHOD;

 ASK METHOD GetTimeInDepot() : REAL;
  BEGIN
    RETURN DepotStats.TimeInDepot;
  END METHOD;

 ASK METHOD GetTour() : INTEGER;
  BEGIN
    RETURN Tour;
  END METHOD;

 ASK METHOD GetWingType() : INTEGER;
  BEGIN
    RETURN WingType;
  END METHOD;

 ASK METHOD InitialData(IN DataString  :  STRING; IN Process : INTEGER);
  CONST
    ProcessASPA = 1;
  VAR
    BunoString, HrsString, LndgsString, CatsString  :  STRING;
    DataReader  :  DataFinderObj;
  BEGIN
    NEW(DataReader);
    AircraftAge := 0.0;
    Buno := ASK DataReader TO GetBuno(DataString);
    WingType := ASK DataReader TO GetWingType(DataString);
    Block := ASK DataReader TO GetBlock();
    Status := ASK DataReader GetStatus();
    Hrs := ASK DataReader GetHrs();
    Lndgs := ASK DataReader GetLndgs();
    Cats := ASK DataReader GetCats();
    FourG := ASK DataReader GetFourG();
    FiveG := ASK DataReader GetFiveG();
    SixG := ASK DataReader GetSixG();
    SevenG := ASK DataReader GetSevenG();
    FLE := ASK DataReader GetFLE();
    ExtHrs := ASK DataReader GetExtHrs();
    Tour := ASK DataReader GetTour();
    ASPA := ASK DataReader GetASPA();
    PED := ASK DataReader GetPED();
    IF Process = ProcessASPA
      OSP := ASK DataReader GetOSP();
    ELSE
      OSP := PDMCycle;
    END IF;
    ASK SELF TO InitialDepotFactors;
    NEW(DepotStats);
    ASK SELF TO InitialDepotStats;
    UpdateFactor := 1.0;
    ASK SELF TO UpdateCumData;
    DISPOSE(DataReader);
  END METHOD;

ASK METHOD InitialDepotFactors;
  BEGIN
    OSPLimit := FALSE;
    PEDLimit := FALSE;
    FLELimit := FALSE;
    EXTLimit := FALSE;
    IF Block <> 3
      NeedBlock := TRUE;
    ELSE
      NeedBlock := FALSE;
    END IF;
  END METHOD;

ASK METHOD InitialDepotStats;
  BEGIN
    DepotStats.NumberOfVisits := 0.0;
    DepotStats.TimeInDepot := 0.0;
    DepotStats.LaborMonths := 0.0;
    DepotStats.LaborHours := 0.0;
    DepotStats.CumTimeInDepot := 0.0;
    DepotStats.CumLaborMonths := 0.0;
    DepotStats.CumLaborHours := 0.0;
    DepotStats.IDT := 0.0;
    DepotExitTime := SimTime();
  END METHOD;

ASK METHOD InitialPDMCycle(IN NumberOfMonths : INTEGER);
  BEGIN
    PDMCycle := NumberOfMonths;
  END METHOD;

ASK METHOD IncrementASPA;
  BEGIN
    INC(ASPA);
  END METHOD;
```

```
 ASK METHOD IncrementTour;
   BEGIN
     INC(Tour);
   END METHOD;

ASK METHOD ReInitializeASPA;
   CONST
     Op = "Op";
   BEGIN
     ASK SELF TO ChangeStatus(Op);
     IF JobList[Rework].TaskRecord
       ASK SELF TO IncrementTour;
       ASPA := 1;
       OSP := 36;
       PED := 36;
       OSPLimit := FALSE;
       PEDLimit := FALSE;
     END IF;
     IF JobList[Rewing].TaskRecord
       FLE := 0.0;
       ExtHrs := 0.0;
       FLELimit := FALSE;
       EXTLimit := FALSE;
     END IF;
     IF JobList[ReworkUpgrade].TaskRecord
       ASK SELF TO IncrementTour;
       ASPA := 1;
       OSP := 36;
       PED := 36;
       OSPLimit := FALSE;
       PEDLimit := FALSE;
       Block := 3;
       NeedBlock := FALSE;
     END IF;
     IF JobList[RewingUpgrade].TaskRecord
       FLE := 0.0;
       ExtHrs := 0.0;
       FLELimit := FALSE;
       EXTLimit := FALSE;
       Block := 3;
       NeedBlock := FALSE;
     END IF;
     IF JobList[ReworkRewing].TaskRecord
       ASK SELF TO IncrementTour;
       ASPA := 1;
       OSP := 36;
       PED := 36;
       OSPLimit := FALSE;
       PEDLimit := FALSE;
       FLE := 0.0;
       ExtHrs := 0.0;
       FLELimit := FALSE;
       EXTLimit := FALSE;
     END IF;
```

```
    IF JobList[ReworkRewingUpgrade].TaskRecord
      ASK SELF TO IncrementTour;
      ASPA := 1;
      OSP := 36;
      PED := 36;
      OSPLimit := FALSE;
      PEDLimit := FALSE;
      FLE := 0.0;
      ExtHrs := 0.0;
      FLELimit := FALSE;
      EXTLimit := FALSE;
      Block := 3;
      NeedBlock := FALSE;
    END IF;
  END METHOD;

ASK METHOD ReInitializePDM;
  CONST
    Op = "Op";
  BEGIN
    ASK SELF TO ChangeStatus(Op);
    IF JobList[Rework].TaskRecord
      ASK SELF TO IncrementTour;
      ASPA := 0;
      OSP := PDMCycle;
      PED := PDMCycle;
      OSPLimit := FALSE;
      PEDLimit := FALSE;
    END IF;
    IF JobList[Rewing].TaskRecord
      FLE := 0.0;
      ExtHrs := 0.0;
      FLELimit := FALSE;
      EXTLimit := FALSE;
    END IF;
    IF JobList[ReworkUpgrade].TaskRecord
      ASK SELF TO IncrementTour;
      ASPA := 0;
      OSP := PDMCycle;
      PED := PDMCycle;
      OSPLimit := FALSE;
      PEDLimit := FALSE;
      Block := 3;
      NeedBlock := FALSE;
    END IF;
    IF JobList[RewingUpgrade].TaskRecord
      FLE := 0.0;
      ExtHrs := 0.0;
      FLELimit := FALSE;
      EXTLimit := FALSE;
      Block := 3;
      NeedBlock := FALSE;
    END IF;
    IF JobList[ReworkRewing].TaskRecord
```

```
      ASK SELF TO IncrementTour;
      ASPA := 0;
      OSP := PDMCycle;
      PED := PDMCycle;
      OSPLimit := FALSE;
      PEDLimit := FALSE;
      FLE := 0.0;
      ExtHrs := 0.0;
      FLELimit := FALSE;
      EXTLimit := FALSE;
    END IF;
    IF JobList[ReworkRewingUpgrade].TaskRecord
      ASK SELF TO IncrementTour;
      ASPA := 0;
      OSP := PDMCycle;
      PED := PDMCycle;
      OSPLimit := FALSE;
      PEDLimit := FALSE;
      FLE := 0.0;
      ExtHrs := 0.0;
      FLELimit := FALSE;
      EXTLimit := FALSE;
      Block := 3;
      NeedBlock := FALSE;
    END IF;
  END METHOD;

ASK METHOD SetDepotStats(IN MMTHS, MMHRS, Cost : REAL);
  BEGIN
    DepotStats.LaborMonths := MMTHS;
    DepotStats.LaborHours := MMHRS;
    DepotStats.TaskCost := Cost;
    DepotStats.CumLaborMonths := DepotStats.CumLaborMonths + MMTHS;
    DepotStats.CumLaborHours := DepotStats.CumLaborHours + MMHRS;
    DepotStats.CumCost := DepotStats.CumCost + Cost;
    DepotStats.NumberOfVisits := DepotStats.NumberOfVisits + 1.0;;
  END METHOD;

ASK METHOD SetExecuteTask(IN TaskCondition : BOOLEAN; IN JobNumber : JobOrderType);
  BEGIN
    JobList[JobNumber].ExecuteTask := TaskCondition;
  END METHOD;

ASK METHOD SetPED;
  BEGIN
    PED := 12;
    PEDLimit := FALSE;
  END METHOD;

ASK METHOD SetTaskRecord(IN RecordCondition : BOOLEAN; IN JobNumber : JobOrderType);
  BEGIN
    JobList[JobNumber].TaskRecord := RecordCondition;
  END METHOD;
```

```
ASK METHOD SetTATs(IN TAT : REAL);
  BEGIN
    DepotStats.TimeInDepot := TAT;
    DepotStats.CumTimeInDepot := DepotStats.CumTimeInDepot + TAT;
  END METHOD;

ASK METHOD SetUpdateFactor;
  VAR
    ReInitTime  :  REAL;
    WholeReInitTime  :  INTEGER;
  BEGIN
    ReInitTime := SimTime();
    WholeReInitTime := TRUNC(ReInitTime);
    UpdateFactor := 1.0 -(ReInitTime - FLOAT(WholeReInitTime));
  END METHOD;

ASK METHOD UpdateAircraftAge;
  BEGIN
    AircraftAge := AircraftAge + 1.0;
  END METHOD;

ASK METHOD UpdateCumData;
  BEGIN
    CumHrs := CumHrs + Hrs;
    CumLndgs := CumLndgs + Lndgs;
    CumCats := CumCats + Cats;
    CumFourG := CumFourG + FourG;
    CumFiveG := CumFiveG + FiveG;
    CumSixG := CumSixG + SixG;
    CumSevenG := CumSevenG + SevenG;
  END METHOD;

ASK METHOD UpdateData(IN Gen  :  FltDataGeneratorObj);
  VAR
    Test  :  REAL;
    Continue  :  CHAR;
    TempLndgs, TempCats, TempFourG, TempFiveG, TempSixG, TempSevenG  :  REAL;
  BEGIN
    IF Status <> "Op"
      OUTPUT("Aircraft in Depot Status is still flying!");
      ASK SELF TO MonthlyReport;
      OUTPUT("Hit RETURN to continue");
      Continue := ReadKey;
    END IF;
    Hrs := UpdateFactor*ASK Gen NewHrs;
    TempLndgs := UpdateFactor*(ASK Gen NewLndgs);
    Lndgs := TRUNC(TempLndgs);
    TempCats := UpdateFactor*(ASK Gen NewCats);
    Cats := TRUNC(TempCats);
    TempFourG := UpdateFactor*(ASK Gen NewFourG);
    FourG := TRUNC(TempFourG);
    TempFiveG := UpdateFactor*(ASK Gen NewFiveG);
    FiveG := TRUNC(TempFiveG);
    TempSixG := UpdateFactor*(ASK Gen NewSixG);
    SixG := TRUNC(TempSixG);
    TempSevenG := UpdateFactor*(ASK Gen NewSevenG);
    SevenG := TRUNC(TempSevenG);
    ASK SELF TO UpdateCumData;
    ASK SELF TO UpdateFLE;
    IF FLELimit
      ExtHrs := UpdateFactor*(ExtHrs + Hrs);
    END IF;
    ASK SELF TO UpdatePeriods;
    ASK SELF TO UpdateDepotFactors;
    UpdateFactor := 1.0;
  END METHOD;

ASK METHOD UpdateDepotExitTime(IN ExitTime : REAL);
  BEGIN
    DepotExitTime := ExitTime;
  END METHOD;

 ASK METHOD UpdateDepotFactors;
  BEGIN
    IF (OSP = 0)
      OSPLimit := TRUE;
    END IF;
    IF (PED = 0)
      PEDLimit := TRUE;
    END IF;
    IF (FLE > 90.0)
      FLELimit := TRUE;
    END IF;
    IF (ExtHrs > 600.0)
      EXTLimit := TRUE;
    END IF;
    IF Block <> 3
      NeedBlock := TRUE;
    END IF;
  END METHOD;

ASK METHOD UpdateDepotStats;
  BEGIN
    DepotStats.CumLaborMonths := DepotStats.CumLaborMonths + DepotStats.LaborMonths;
    DepotStats.CumLaborHours := DepotStats.CumLaborHours + DepotStats.LaborHours;
    DepotStats.CumCost := DepotStats.CumCost + DepotStats.TaskCost;
  END METHOD;

ASK METHOD UpdateFLE;
  VAR
    C4,C5,C6,C7,CL,CC  :  REAL;
  BEGIN
    IF WingType = 7050
      C4 := 0.020051;
      C5 := 0.05863;
      C6 := 0.145304;
      C7 := 0.288043;
      CL := 0.001894;
      CC := 0.005808;
    END IF;
    IF WingType = 7075
      C4 := 0.006884;
      C5 := 0.043298;
      C6 := 0.196077;
      C7 := 0.282965;
      CL := 0.00098;
      CC := 0.001562;
    END IF;
    FLE := FLE + UpdateFactor*(C4*FLOAT(FourG-FiveG)+C5*FLOAT(FiveG-SixG)
              +C6*FLOAT(SixG-SevenG)+(C7*FLOAT(SevenG)) +(CL*FLOAT(Lndgs))
              +(CC*FLOAT(Cats)));
  END METHOD;

ASK METHOD UpdateIDT(IN EntryTime : REAL);
  BEGIN
    DepotStats.IDT := EntryTime - DepotExitTime;
  END METHOD;

ASK METHOD UpdatePeriods;
  BEGIN
    DEC(OSP);
    DEC(PED);
  END METHOD;

ASK METHOD UpdateStats(IN StatRecord : StatsObj; IN YearlyStatRecord : YearlyStatsObj);
  BEGIN
    ASK StatRecord TO GetSample((ASK SELF TO GetTimeInDepot),
                                (ASK SELF TO GetLaborMonths),
                                (ASK SELF TO GetLaborHours));
    ASK YearlyStatRecord TO GetSample(ASK SELF TO GetLaborHours);
  END METHOD;

TELL METHOD GoToDepot(IN RunNumber, InductionProcess : INTEGER;
                      IN Generator : FltDataGeneratorObj;
                      IN Depot : DepotObj; IN StatRecord : StatsObj;
                      IN YearlyStatRecord : YearlyStatsObj);
  CONST
    Op = "Op";
    ProcessASPA = 1;
  VAR
    Manager  :  DataManagerObj;
    EntryTime, TurnAroundTime :  REAL;
    Continue : CHAR;
  BEGIN
    NEW(Manager);
    ASK SELF TO UpdateIDT(SimTime());
    EntryTime := SimTime();
    WAIT FOR Depot TO Give(SELF,1);
    END WAIT;
    WAIT FOR Depot TO ProcessAircraft(SELF);
    END WAIT;
    ASK Depot TO TakeBack(SELF,1);
```

```
      TurnAroundTime := SimTime() - EntryTime;
      ASK SELF TO SetTATs(TurnAroundTime);
      ASK SELF TO UpdateStats(StatRecord,YearlyStatRecord);
      ASK SELF TO ChangeStatus(Op);
      ASK SELF TO SetUpdateFactor;
      ASK Generator TO GenUpdateData;
      ASK SELF TO UpdateData(Generator);
      IF InductionProcess = ProcessASPA
        ASK SELF TO ReInitializeASPA;
      ELSE
        ASK SELF TO ReInitializePDM;
      END IF;
      ASK SELF TO UpdateDepotExitTime(SimTime());
      DISPOSE(Manager);
    END METHOD;

END OBJECT;

END MODULE.
```

```
DEFINITION MODULE Operations;

FROM DataManager IMPORT DataManagerObj;
FROM Depot IMPORT DepotObj;
FROM GrpMod IMPORT StatQueueObj;
FROM NumberMill IMPORT FltDataGeneratorObj, JobDataGeneratorObj;
FROM Screen IMPORT ScreenObj;
FROM Stats IMPORT StatsObj, CumStatsObj, YearlyStatsObj;

TYPE
  OperationsObj = OBJECT
    AllAircraft  :  StatQueueObj;
    DepotOne  : DepotObj;
    FlightDataGenerator  :  FltDataGeneratorObj;
    InspectionTeam  :  ScreenObj;
    JobGenerator  :  JobDataGeneratorObj;
    Manager  :  DataManagerObj;
    ASK METHOD ObjInit;
    ASK METHOD PrepareReports(IN Process : INTEGER; IN FileName : STRING);
    TELL METHOD MonthlyOps(IN FileName : STRING; IN RunNumber, PDMCycle : INTEGER;
                           IN RunLength : REAL; IN Process : INTEGER;
                           IN StatRecord  : StatsObj; IN CumStatRecord : CumStatsObj;
                           IN YearlyStatRecord : YearlyStatsObj);
    ASK METHOD YearMark(IN Time : REAL) : BOOLEAN;
    ASK METHOD ObjTerminate;
END OBJECT;

END MODULE.
```

```
IMPLEMENTATION MODULE Operations;

FROM Aircraft IMPORT AircraftObj;
FROM DataManager IMPORT DataManagerObj;
FROM Depot IMPORT DepotManagerObj, DepotObj;
FROM IOMod IMPORT StreamObj, FileUseType(Append);
FROM NumberMill IMPORT FltDataGeneratorObj, JobDataGeneratorObj;
FROM Screen IMPORT ScreenObj;
FROM SimMod IMPORT SimTime;
FROM Stats IMPORT StatsObj, CumStatsObj, YearlyStatsObj;

OBJECT OperationsObj;
  ASK METHOD ObjInit;
    BEGIN
      NEW(Manager);
      NEW(FlightDataGenerator);
      NEW(InspectionTeam);
      NEW(JobGenerator);
    END METHOD;

  ASK METHOD PrepareReports(IN Process : INTEGER; IN FileName : STRING);
    BEGIN
      ASK Manager TO InitializeReports(Process,FileName);
    END METHOD;

  TELL METHOD MonthlyOps(IN FileName : STRING; IN RunNumber, PDMCycle : INTEGER;
                         IN RunLength : REAL; IN Process : INTEGER;
                         IN StatRecord  : StatsObj; IN CumStatRecord : CumStatsObj;
                         IN YearlyStatRecord : YearlyStatsObj);
    CONST
      Op = "Op";
      Depot = "Depot";
      ASPA = 1;
      PDM = 2;
    VAR
      Lot  :  REAL;
      SendToDepot  :  BOOLEAN;
      LotNumber  :  STRING;
      Prowler  :  AircraftObj;
      DataWriter  :  StreamObj;
    BEGIN
      NEW(AllAircraft);
      NEW(DepotOne);
      ASK DepotOne TO SetJobDataGen(JobGenerator);
      WHILE SimTime() < RunLength;
        WAIT DURATION 1.0
          IF (ASK Manager TO GetDeliveryTime(SimTime(),LotNumber));
            ASK Manager TO InitializeAircraft(AllAircraft,LotNumber,Process,PDMCycle);
          END IF;
          IF (SimTime() = 1.0) OR (ASK SELF TO YearMark(SimTime()))
            WAIT FOR  DepotOne TO UpdateDepotAvailability(AllAircraft,Process);
            END WAIT;
          END IF;
          FOREACH Prowler IN AllAircraft
            ASK Prowler TO UpdateAircraftAge;
            IF (ASK Prowler TO CheckStatus) = Op
              ASK FlightDataGenerator TO GenUpdateData;
              ASK Prowler TO UpdateData(FlightDataGenerator);
              ASK InspectionTeam TO DepotScreen(Prowler,Process,SendToDepot);
              IF SendToDepot
                ASK Prowler TO ChangeStatus(Depot);
                TELL Prowler TO GoToDepot(RunNumber,Process,FlightDataGenerator,DepotOne,
                                                  StatRecord,YearlyStatRecord);
              END IF;
            END IF;
          END FOREACH;
          IF ASK SELF TO YearMark(SimTime())
            ASK YearlyStatRecord TO GetYearlySample;
          END IF;
        END WAIT;
      END WHILE;
      FOREACH Prowler IN AllAircraft
        ASK CumStatRecord TO GetCumSample(Prowler);
      END FOREACH;
      ASK Manager TO StatReport(FileName,RunNumber,Process,
                           (ASK StatRecord TO MeanTAT),(ASK StatRecord TO StdDevTAT),
                           (ASK StatRecord TO MeanMMTHS),(ASK StatRecord TO StdDevTAT),
                           (ASK StatRecord TO MeanMMHRS),(ASK StatRecord TO StdDevMMHRS),
                           (ASK CumStatRecord TO MeanCumTime),
                           (ASK CumStatRecord TO StdDevCumTime),
                           (ASK CumStatRecord TO MeanVisits),(ASK CumStatRecord TO StdDevVisits),
                           (ASK StatRecord TO MeanMMHRSYr(RunLength/12.0)),
                           (ASK YearlyStatRecord TO GetYearlyMeanMMHRS),
                           (ASK YearlyStatRecord TO GetYearlyStdDevMMHRS),
                           (ASK YearlyStatRecord TO GetYearlyMeanVisits),
                           (ASK YearlyStatRecord TO GetYearlyStdDevVisits));
    END METHOD;

  ASK METHOD YearMark(IN Time : REAL) : BOOLEAN;
    VAR
      TempTime  :  INTEGER;
    BEGIN
      TempTime := TRUNC(Time/12.0);
      IF (FLOAT(TempTime) - Time/12.0) = 0.0
        RETURN TRUE;
      ELSE
        RETURN FALSE;
      END IF;
    END METHOD;

  ASK METHOD ObjTerminate;
    BEGIN
      DISPOSE(Manager);
    END METHOD;

END OBJECT;
END MODULE.
```

```
DEFINITION MODULE Screen;

FROM Aircraft IMPORT AircraftObj;
FROM RandMod IMPORT RandomObj;

TYPE
  ScreenObj = OBJECT;
  ASPATeam  : InspectorObj;
  ASK METHOD ObjInit;
  ASK METHOD DepotScreen(IN Prowler : AircraftObj; IN Process : INTEGER;
                         OUT SendDepot : BOOLEAN);
  ASK METHOD DepotScreenASPA(IN Prowler : AircraftObj;OUT SendToDepot : BOOLEAN);
  ASK METHOD DepotScreenPDM(IN Prowler : AircraftObj;OUT SendToDepot : BOOLEAN);
  ASK METHOD GetWorkOrder(IN Jet : AircraftObj; IN Rework,Rewing,Upgrade : BOOLEAN) :
                                     INTEGER;
  END OBJECT;

  InspectorObj = OBJECT;
    RandScore  :  RandomObj;
    ASK METHOD ObjInit;
    ASK METHOD InspectASPA(IN ASPANumber : INTEGER;OUT Pass : BOOLEAN);
  END OBJECT;

END MODULE.
```

```
IMPLEMENTATION MODULE Screen;

FROM Aircraft IMPORT AircraftObj,JobOrderType;
FROM IOMod IMPORT ReadKey;
FROM RandMod IMPORT RandomObj, FetchSeed;

OBJECT ScreenObj;
  ASK METHOD ObjInit;
      BEGIN
        NEW(ASPATeam);
      END METHOD;

  ASK METHOD DepotScreen(IN Prowler : AircraftObj; IN Process : INTEGER; OUT SendDepot :
                                                BOOLEAN);
    CONST
      ASPA = 1;
      PDM = 2;
    BEGIN
      IF Process = ASPA
        ASK SELF TO DepotScreenASPA(Prowler,SendDepot);
      ELSE
        ASK SELF TO DepotScreenPDM(Prowler,SendDepot);
      END IF;
    END METHOD;

  ASK METHOD DepotScreenASPA(IN Prowler : AircraftObj;OUT SendToDepot : BOOLEAN);
    VAR
      JobNumber  :  JobOrderType;
      ASPANumber, OrderNumber  :  INTEGER;
      PassASPA, TaskCondition, Rework, Rewing, Upgrade  :  BOOLEAN;
      Continue  :  CHAR;
    BEGIN
      IF ASK Prowler TO CheckStatus <> "Op"
        OUTPUT("Aircraft already in Depot Status is being screened!");
        ASK Prowler TO MonthlyReport;
        OUTPUT("Hit RETURN to continue");
        Continue := ReadKey;
      END IF;
      ASPANumber := ASK Prowler TO GetASPA;
      TaskCondition := FALSE;
      SendToDepot := FALSE;
      FOR JobNumber := MIN(JobOrderType) TO MAX(JobOrderType)
        ASK Prowler TO SetExecuteTask(TaskCondition, JobNumber);
        ASK Prowler TO SetTaskRecord(TaskCondition, JobNumber);
      END FOR;
      IF (ASK Prowler TO GetOSPLimit) AND (ASPANumber = 1);
        ASK ASPATeam TO InspectASPA(ASPANumber,PassASPA);
        IF PassASPA
          ASK Prowler TO IncrementASPA;
          ASK Prowler TO SetPED;
        ELSE
          Rework := TRUE;
          SendToDepot := TRUE;
          TaskCondition := TRUE;
        END IF;
      END IF;
      IF (ASK Prowler TO GetPEDLimit) AND (ASK Prowler TO GetOSPLimit)
        ASK ASPATeam TO InspectASPA(ASPANumber,PassASPA);
        IF PassASPA
          ASK Prowler TO IncrementASPA;
          ASK Prowler TO SetPED;
        ELSE
          Rework := TRUE;
          SendToDepot := TRUE;
          TaskCondition := TRUE;
        END IF;
      END IF;
      IF ASK Prowler TO GetFLELimit AND ASK Prowler TO GetEXTLimit
        IF ASPANumber = 4
          Rework := TRUE;
        END IF;
        Rewing := TRUE;
        SendToDepot := TRUE;
        TaskCondition := TRUE;
      END IF;
      IF ASK Prowler TO GetNeedBlock
        Upgrade := TRUE;
      END IF;
      OrderNumber := ASK SELF TO GetWorkOrder(Prowler,Rework,Rewing,Upgrade);
      CASE OrderNumber
        WHEN 0:
          ASK Prowler TO SetExecuteTask(TaskCondition,VAL(JobOrderType,OrderNumber));
        WHEN 1:
          ASK Prowler TO SetExecuteTask(TaskCondition,VAL(JobOrderType,OrderNumber));
        WHEN 2:
          ASK Prowler TO SetExecuteTask(TaskCondition,VAL(JobOrderType,OrderNumber));
        WHEN 3:
          ASK Prowler TO SetExecuteTask(TaskCondition,VAL(JobOrderType,OrderNumber));
        WHEN 4:
          ASK Prowler TO SetExecuteTask(TaskCondition,VAL(JobOrderType,OrderNumber));
        WHEN 5:
          ASK Prowler TO SetExecuteTask(TaskCondition,VAL(JobOrderType,OrderNumber));
        OTHERWISE
      END CASE;
    END METHOD;

  ASK METHOD DepotScreenPDM(IN Prowler : AircraftObj; OUT SendToDepot : BOOLEAN);
    VAR
      JobNumber  :  JobOrderType;
      OrderNumber  :  INTEGER;
      PassASPA, TaskCondition, Rework, Rewing, Upgrade  :  BOOLEAN;
      Continue  :  CHAR;
    BEGIN
      IF ASK Prowler TO CheckStatus <> "Op"
        OUTPUT("Aircraft already in Depot Status is being screened!");
        ASK Prowler TO MonthlyReport;
        OUTPUT("Hit RETURN to continue");
        Continue := ReadKey;
      END IF;
      TaskCondition := FALSE;
      SendToDepot := FALSE;
      FOR JobNumber := MIN(JobOrderType) TO MAX(JobOrderType)
        ASK Prowler TO SetExecuteTask(TaskCondition, JobNumber);
        ASK Prowler TO SetTaskRecord(TaskCondition, JobNumber);
      END FOR;
      IF ASK Prowler TO GetOSPLimit;
          Rework := TRUE;
          SendToDepot := TRUE;
          TaskCondition := TRUE;
      END IF;
      IF ASK Prowler TO GetFLELimit AND ASK Prowler TO GetEXTLimit
        Rewing := TRUE;
        SendToDepot := TRUE;
        TaskCondition := TRUE;
      END IF;
      IF ASK Prowler TO GetNeedBlock
        Upgrade := TRUE;
      END IF;
      OrderNumber := ASK SELF TO GetWorkOrder(Prowler,Rework,Rewing,Upgrade);
      CASE OrderNumber
        WHEN 0:
          ASK Prowler TO SetExecuteTask(TaskCondition,VAL(JobOrderType,OrderNumber));
        WHEN 1:
          ASK Prowler TO SetExecuteTask(TaskCondition,VAL(JobOrderType,OrderNumber));
        WHEN 2:
          ASK Prowler TO SetExecuteTask(TaskCondition,VAL(JobOrderType,OrderNumber));
        WHEN 3:
          ASK Prowler TO SetExecuteTask(TaskCondition,VAL(JobOrderType,OrderNumber));
        WHEN 4:
          ASK Prowler TO SetExecuteTask(TaskCondition,VAL(JobOrderType,OrderNumber));
        WHEN 5:
          ASK Prowler TO SetExecuteTask(TaskCondition,VAL(JobOrderType,OrderNumber));
        OTHERWISE
      END CASE;
    END METHOD;

  ASK METHOD GetWorkOrder(IN Jet : AircraftObj; IN Rework,Rewing,Upgrade : BOOLEAN) :
                                          INTEGER;
    VAR
      WorkOrder  :  INTEGER;
    BEGIN
      IF  Rework
        WorkOrder := 0;
      END IF;
      IF Rewing
        WorkOrder := 1;
      END IF;
      IF Rework AND Rewing
        WorkOrder := 2;
      END IF;
      IF Rework AND Upgrade
        WorkOrder := 3;
      END IF;
      IF Rewing AND Upgrade
        WorkOrder := 4;
      END IF;
      IF Rework AND Rewing AND Upgrade
        WorkOrder := 5;
      END IF;
      RETURN WorkOrder;
    END METHOD;
END OBJECT;

OBJECT InspectorObj;
  ASK METHOD ObjInit;
    BEGIN
      NEW(RandScore);
      ASK RandScore TO SetSeed(FetchSeed(8));
    END METHOD;

  ASK METHOD InspectASPA(IN ASPANumber : INTEGER;OUT Pass : BOOLEAN);
    VAR
      Score  :  REAL;
    BEGIN
      Score := ASK RandScore TO Sample;
      CASE ASPANumber
        WHEN 1:
          IF Score  < 0.0305
            Pass := FALSE;
          ELSE
            Pass := TRUE;
          END IF;
        WHEN 2:
          IF Score < 0.1048
            Pass := FALSE;
          ELSE
            Pass := TRUE;
          END IF;
        WHEN 3:
          IF Score < 0.1471
            Pass := FALSE;
          ELSE
            Pass := TRUE;
          END IF;
        WHEN 4:
          IF Score < 0.4255
            Pass := FALSE;
          ELSE
            Pass := TRUE;
          END IF;
        WHEN 5:
          IF Score < 0.6000
            Pass := FALSE;
          ELSE
            Pass := TRUE;
          END IF;
        OTHERWISE
          Pass := FALSE;
      END CASE;
    END METHOD;
END OBJECT;

END MODULE.
```

```
DEFINITION MODULE Depot;

FROM Aircraft IMPORT AircraftObj;
FROM GrpMod IMPORT StatQueueObj;
FROM NumberMill IMPORT JobDataGeneratorObj;
FROM ResMod IMPORT ResourceObj;

TYPE
  DepotObj = OBJECT(ResourceObj);
    DepotSlots  :  INTEGER;
    DepotManager  :  DepotManagerObj;
    JobDataGen  :  JobDataGeneratorObj;
    ASK METHOD GetAvailability() : INTEGER;
    ASK METHOD GetCapacity() : INTEGER;
    ASK METHOD GetNonAvailability() : INTEGER;
    ASK METHOD GetJobTimes(IN Jet : AircraftObj; IN JobName : STRING);
    WAITFOR METHOD ProcessAircraft(IN Prowler : AircraftObj);
    ASK METHOD SetJobDataGen(IN JobGenerator : JobDataGeneratorObj);
    WAITFOR  METHOD UpdateDepotAvailability(IN AllAircfraft : StatQueueObj; IN Process :
                                                              INTEGER);
    OVERRIDE
      ASK METHOD ObjInit;
  END OBJECT;

  DepotManagerObj = OBJECT;
    ASK METHOD GetDepotAvailabilityASPA(IN AllAircraft : StatQueueObj) : INTEGER;
    ASK METHOD GetDepotAvailabilityPDM(IN AllAircraft : StatQueueObj) : INTEGER;
    ASK METHOD GetMaxTour(IN AllAircraft : StatQueueObj) : INTEGER;
  END OBJECT;

END MODULE.
```

```
IMPLEMENTATION MODULE Depot;

FROM Aircraft IMPORT AircraftObj, JobOrderType;
FROM DataManager IMPORT DataManagerObj;
FROM GrpMod IMPORT StatQueueObj;
FROM MathMod IMPORT CEIL;
FROM NumberMill IMPORT JobDataGeneratorObj;

OBJECT DepotObj;
  ASK METHOD ObjInit;
    BEGIN
      INHERITED ObjInit;
      DepotSlots := 0;
      ASK SELF TO Create(DepotSlots);
      NEW(DepotManager);
    END METHOD;

  ASK METHOD SetJobDataGen(IN JobGenerator : JobDataGeneratorObj);
    BEGIN
      JobDataGen := JobGenerator;
    END METHOD;

  ASK METHOD GetAvailability() : INTEGER;
    BEGIN
      RETURN (ASK SELF TO ReportAvailability);
    END METHOD;

  ASK METHOD GetCapacity() : INTEGER;
    BEGIN
      RETURN (ASK SELF MaxResources);
    END METHOD;

  ASK METHOD GetJobTimes(IN Jet : AircraftObj; IN JobName : STRING);
    VAR
      ManMths, ManHours, Price  :  REAL;
    BEGIN
      CASE JobName
        WHEN "Rework":
          ASK JobDataGen TO GetReworkTimes((ASK Jet TO GetIDT),ManMths,ManHours,Price);
          ASK Jet TO SetDepotStats(ManMths,ManHours,Price);
        WHEN "Rewing":
          ASK JobDataGen TO GetRewingTimes(ManMths,ManHours, Price);
          ASK Jet TO SetDepotStats(ManMths, ManHours, Price);
        WHEN "ReworkUpgrade":
          ASK JobDataGen TO GetReworkUpgradeTimes((ASK Jet TO GetIDT),ManMths,
                                                   ManHours, Price);
          ASK Jet TO SetDepotStats(ManMths, ManHours, Price);
        WHEN "RewingUpgrade":
          ASK JobDataGen TO GetRewingUpgradeTimes(ManMths,ManHours, Price);
          ASK Jet TO SetDepotStats(ManMths, ManHours, Price);
        WHEN "ReworkRewing":
          ASK JobDataGen TO GetReworkRewingTimes((ASK Jet TO GetIDT),ManMths,
                                                  ManHours, Price);
          ASK Jet TO SetDepotStats(ManMths, ManHours, Price);
        WHEN "ReworkRewingUpgrade":
          ASK JobDataGen TO GetReworkRewingUpgradeTimes((ASK Jet TO GetIDT),
                                                         ManMths,ManHours, Price);
          ASK Jet TO SetDepotStats(ManMths, ManHours, Price);
        OTHERWISE
          ManMths := 0.0;
          ManHours := 0.0;
          Price := 0.0;
      END CASE;
    END METHOD;

  ASK METHOD GetNonAvailability() : INTEGER;
    BEGIN
      RETURN (ASK SELF TO NumberAllocatedTo(SELF));
    END METHOD;

  WAITFOR METHOD ProcessAircraft(IN Prowler : AircraftObj);
    VAR
      Count  :  INTEGER;
      WaitTime  :  REAL;
      TaskCondition  :  BOOLEAN;
      JobNumber  :  JobOrderType;
    BEGIN
      TaskCondition := FALSE;
      FOR JobNumber := MIN(JobOrderType) TO MAX(JobOrderType)
        IF (ASK Prowler TO GetExecuteTask(JobNumber))
          TaskCondition := TRUE;
          ASK SELF TO GetJobTimes(Prowler,ASK Prowler TO GetTaskName(JobNumber));
          WaitTime := ASK Prowler TO GetLaborMonths;
          WAIT DURATION WaitTime;
          END WAIT;
          ASK Prowler TO SetTaskRecord(TaskCondition,VAL(JobOrderType,ORD(JobNumber)));
        END IF;
      END FOR;
    END METHOD;

  WAITFOR METHOD UpdateDepotAvailability(IN AllAircraft : StatQueueObj; IN Process : INTEGER);
    CONST
      ASPA = 1;
      PDM = 2;
    VAR
      NewDepotSlots  :  INTEGER;
      Continue : CHAR;
    BEGIN
      IF Process = ASPA
        NewDepotSlots := ASK DepotManager TO GetDepotAvailabilityASPA(AllAircraft);
      ELSE
        NewDepotSlots := ASK DepotManager TO GetDepotAvailabilityPDM(AllAircraft);
      END IF;
      IF NewDepotSlots > ASK SELF TO ReportAvailability
        ASK SELF IncrementResourcesBy(NewDepotSlots - (ASK SELF TO ReportAvailability));
      ELSE
        IF NewDepotSlots  < ASK SELF TO ReportAvailability
          WAIT FOR SELF DecrementResourcesBy((ASK SELF TO ReportAvailability) - NewDepotSlots);
```

```
          END WAIT;
        END IF;
        IF NewDepotSlots = ASK SELF TO ReportAvailability
          WAIT FOR SELF DecrementResourcesBy((ASK SELF TO ReportAvailability) - NewDepotSlots);
          END WAIT;
        END IF;
      END IF;
    END METHOD;

END OBJECT;

OBJECT DepotManagerObj;
  ASK METHOD GetDepotAvailabilityASPA(IN AllAircraft : StatQueueObj) : INTEGER;
    CONST
      Op = "Op";
    VAR
       MaxTourCount,TourCount,TourCountASPA1,TourCountASPA2,TourCountASPA3,
       TourCountASPA4,TourCountASPA5,Misc,Tour1Slots,Tour2Slots,Tour3Slots,Tour4Slots,
       Tour5Slots,MiscSlots,Slots  :  INTEGER;
       PT1A1,PT1A2,PT1A3,PT1A4,PT1A5,PT2A1,PT2A2,PT2A3,PT2A4,PT2A5,PT3A1,PT3A2,
       PT3A3,PT3A4,PT3A5, PT4A1,PT4A2,PT4A3,PT4A4,PT4A5,PT5A1,PT5A2,PT5A3, PT5A4,
       PT5A5,PTMA1,PTMA2,PTMA3,PTMA4,PTMA5  :  REAL;
      Prowler  :  AircraftObj;
      Manager  :  DataManagerObj;
    BEGIN
      NEW(Manager);
      ASK Manager TO GetRates(PT1A1,PT1A2,PT1A3,PT1A4,PT1A5,PT2A1,PT2A2,PT2A3,
                              PT2A4,PT2A5,PT3A1,PT3A2,PT3A3,PT3A4,PT3A5,PT4A1,
                              PT4A2,PT4A3,PT4A4,PT4A5,PT5A1,PT5A2,PT5A3,PT5A4,
                              PT5A5, PTMA1,PTMA2,PTMA3,PTMA4,PTMA5);
      Slots := 0;
      MaxTourCount := ASK SELF TO GetMaxTour(AllAircraft);
      TourCount := 1;
      FOR TourCount := 1 TO MaxTourCount
        FOREACH Prowler IN AllAircraft
          IF ASK Prowler TO CheckStatus = Op
            IF (ASK Prowler TO GetTour) = TourCount;
              CASE (ASK Prowler TO GetASPA)
                WHEN 1:
                  INC(TourCountASPA1);
                WHEN 2:
                  INC(TourCountASPA2);
                WHEN 3:
                  INC(TourCountASPA3);
                WHEN 4:
                  INC(TourCountASPA4);
                WHEN 5:
                  INC(TourCountASPA5);
                OTHERWISE
                  INC(Misc);
              END CASE;
            END IF;
          END IF;
        END FOREACH;
      CASE TourCount
        WHEN 1:
          Tour1Slots := CEIL(PT1A1*FLOAT(TourCountASPA1) + PT1A2*FLOAT(TourCountASPA2)+
                             PT1A3*FLOAT(TourCountASPA3) + PT1A4*FLOAT(TourCountASPA4) +
                             PT1A5*FLOAT(TourCountASPA5)) + Misc;
          Slots := Slots + Tour1Slots;
        WHEN 2:
          Tour2Slots := CEIL(PT2A1*FLOAT(TourCountASPA1) + PT2A2*FLOAT(TourCountASPA2) +
                            PT2A3*FLOAT(TourCountASPA3+ PT2A4*FLOAT(TourCountASPA4) +
                            PT2A5*FLOAT(TourCountASPA5))+Misc;
           Slots := Slots + Tour2Slots;
        WHEN 3:
          Tour3Slots := CEIL(PT3A1*FLOAT(TourCountASPA1) + PT3A2*FLOAT(TourCountASPA2) +
                            PT3A3*FLOAT(TourCountASPA3+ PT3A4*FLOAT(TourCountASPA4) +
                            PT3A5*FLOAT(TourCountASPA5))+Misc;
          Slots := Slots + Tour3Slots;
        WHEN 4:
          Tour4Slots := CEIL(PT4A1*FLOAT(TourCountASPA1) + PT4A2*FLOAT(TourCountASPA2) +
                             PT4A3*FLOAT(TourCountASPA3) + PT4A4*FLOAT(TourCountASPA4) +
                             PT4A5*FLOAT(TourCountASPA5))+Misc;
          Slots := Slots + Tour4Slots;
        WHEN 5:
          Tour5Slots := CEIL(PT5A1*FLOAT(TourCountASPA1) + PT5A2*FLOAT(TourCountASPA2) +
                             PT5A3*FLOAT(TourCountASPA3) + PT5A4*FLOAT(TourCountASPA4) +
                             PT5A5*FLOAT(TourCountASPA5))+Misc;
          Slots := Slots + Tour5Slots;
        OTHERWISE
          MiscSlots := CEIL(PTMA1*FLOAT(TourCountASPA1) + PTMA2*FLOAT(TourCountASPA2)
                           + PTMA3*FLOAT(TourCountASPA3+ PTMA4*FLOAT(TourCountASPA4)
                           + PTMA5*FLOAT(TourCountASPA5))+Misc;
          Slots := Slots + MiscSlots;
      END CASE;
      INC(TourCount);
      TourCountASPA1 := 0;
      TourCountASPA2 := 0;
      TourCountASPA3 := 0;
      TourCountASPA4 := 0;
      TourCountASPA5 := 0;
      Misc := 0;
    END FOR;
    RETURN Slots;
    DISPOSE(Manager);
  END METHOD;

ASK METHOD GetDepotAvailabilityPDM(IN AllAircraft : StatQueueObj) : INTEGER;
  CONST
    Op = "Op";
  VAR
    Slots  :  INTEGER;
    Prowler  :  AircraftObj;
  BEGIN
    Slots := 0;
    FOREACH Prowler IN AllAircraft
      IF (ASK Prowler TO GetPED < 12) AND (ASK Prowler TO CheckStatus = Op)
```

```
          INC(Slots);
        END IF;
      END FOREACH;
      RETURN Slots;
    END METHOD;

  ASK METHOD GetMaxTour(IN AllAircraft : StatQueueObj) : INTEGER;
    VAR
      MaxTour  :  INTEGER;
      Prowler  :  AircraftObj;
    BEGIN
      MaxTour := 1;
      FOREACH Prowler IN AllAircraft
        IF ASK Prowler TO GetTour > MaxTour
          MaxTour := ASK Prowler TO GetTour;
        END IF;
      END FOREACH;
      RETURN MaxTour;
    END METHOD;
END OBJECT;

END MODULE.
```

```
DEFINITION MODULE NumberMill;

FROM RandMod IMPORT RandomObj;

TYPE
  FltDataGeneratorObj = OBJECT;
    NewLndgs, NewCats,NewFourG,NewFiveG,NewSixG,NewSevenG,NewHrs,
    MeanHrs, StdDevHrs,BetaLndgs, AlphaLndgs,BetaCats, AlphaCats,
    MeanFourG, AlphaFourG,Alpha1FiveG, Alpha2FiveG,StdDevSixG, MeanSixG,
    StdDevSevG, MeanSevG  :  REAL;
    ASK METHOD ObjInit;
    ASK METHOD GenUpdateData;
      RandHrs,RandLndgs,RandCats,RandFourG,
      RandFiveG,RandSixG,RandSevenG  :  RandomObj;
  END OBJECT;

  JobDataGeneratorObj = OBJECT;
    IDTAlpha1, IDTAlpha2, StdError1,
    StdError2, StdError3  :  REAL;
    ASK METHOD ObjInit;
    ASK METHOD GetReworkTimes(IN IDT : REAL; OUT MMTHS, MMHRS, Cost : REAL);
    ASK METHOD GetRewingTimes(OUT MMTHS,MMHRS,Cost : REAL);
    ASK METHOD GetReworkUpgradeTimes(IN IDT : REAL; OUT MMTHS, MMHRS, Cost : REAL);
    ASK METHOD GetRewingUpgradeTimes(OUT MMTHS,MMHRS,Cost : REAL);
    ASK METHOD GetReworkRewingTimes(IN IDT : REAL; OUT MMTHS, MMHRS, Cost : REAL);
    ASK METHOD GetReworkRewingUpgradeTimes(IN IDT : REAL;
                                           OUT MMTHS, MMHRS, Cost : REAL);
    PRIVATE
      RandTime  :  RandomObj;
  END OBJECT;

END MODULE.
```

```
IMPLEMENTATION MODULE NumberMill;

FROM DataManager IMPORT DataManagerObj;
FROM IOMod IMPORT StreamObj, FileUseType(Input);
FROM MathMod IMPORT POWER;
FROM RandMod IMPORT FetchSeed;

OBJECT FltDataGeneratorObj;
  ASK METHOD ObjInit;
    BEGIN
      NEW(RandHrs);
      ASK RandHrs TO SetSeed(FetchSeed(1));
      NEW(RandLndgs);
      ASK RandLndgs TO SetSeed(FetchSeed(2));
      NEW(RandCats);
      ASK RandCats TO SetSeed(FetchSeed(3));
      NEW(RandFourG);
      ASK RandFourG TO SetSeed(FetchSeed(4));
      NEW(RandFiveG);
      ASK RandFiveG TO SetSeed(FetchSeed(5));
      NEW(RandSixG);
      ASK RandSixG TO SetSeed(FetchSeed(6));
      NEW(RandSevenG);
      ASK RandSevenG TO SetSeed(FetchSeed(7));
    END METHOD;

  ASK METHOD GenUpdateData;
    CONST
      P6G = 0.12758;
      P7G = 0.046904;
    VAR
      Prob6G, Prob7G, Tester  :  REAL;
      Manager  :  DataManagerObj;
    BEGIN
      NEW(Manager);
      ASK Manager TO GetFlightParams(MeanHrs,StdDevHrs,AlphaLndgs,BetaLndgs,AlphaCats,
                                                       BetaCats,MeanFourG,AlphaFourG,Alpha1FiveG,Alpha2FiveG);
      REPEAT
        NewHrs := ASK RandHrs TO Normal(MeanHrs,StdDevHrs);
      UNTIL NewHrs > 0.0;
      NewLndgs := ASK RandLndgs TO Weibull(AlphaLndgs,BetaLndgs);
      NewCats := ASK RandCats TO Weibull(AlphaCats,BetaCats);
      NewFourG := ASK RandFourG TO Gamma(MeanFourG,AlphaFourG);
      NewFiveG := -0.5 +53.0 * (ASK RandFiveG TO Beta(Alpha1FiveG,Alpha2FiveG));
      IF ASK RandSixG TO Sample < P6G
        NewSixG := 1.0;
      ELSE
        NewSixG := 0.0;
      END IF;
      IF ASK RandSevenG TO Sample < P7G
        NewSevenG := 1.0;
      ELSE
        NewSevenG := 0.0;
      END IF;
      DISPOSE(Manager);
    END METHOD;
END OBJECT;

OBJECT JobDataGeneratorObj;
  ASK METHOD ObjInit;
    VAR
      ParamFinder  :  StreamObj;
      SpaceString  :  STRING;
    BEGIN
      NEW(RandTime);
      ASK RandTime TO SetSeed(FetchSeed(9));
    END METHOD;

  ASK METHOD GetReworkTimes(IN IDT : REAL; OUT MMTHS, MMHRS, Cost : REAL);
    VAR
      Epsilon1,Epsilon2,Epsilon3 : REAL;
      Manager  :  DataManagerObj;
    BEGIN
      NEW(Manager);
      ASK Manager TO GetJobParams(StdError1,StdError2,StdError3);
      IDT := IDT * 30.0;
      REPEAT
        Epsilon1 := ASK RandTime TO Normal(0.0,StdError1);
        MMHRS := 6.4297*IDT + Epsilon1;
      UNTIL MMHRS > 0.0;
      MMTHS := (0.0136 * MMHRS) * 0.0333;
      DISPOSE(Manager);
    END METHOD;

  ASK METHOD GetRewingTimes(OUT MMTHS,MMHRS,Cost : REAL);
    BEGIN
      MMTHS := 4.0;
      MMHRS := MMTHS/(0.0136*0.033);
    END METHOD;

  ASK METHOD GetReworkUpgradeTimes(IN IDT : REAL; OUT MMTHS, MMHRS, Cost : REAL);
    BEGIN
      ASK SELF TO GetReworkTimes(IDT,MMTHS,MMHRS,Cost);
      MMTHS := (MMHRS * 0.0136 * 0.033) + 0.5;
    END METHOD;

  ASK METHOD GetRewingUpgradeTimes(OUT MMTHS,MMHRS,Cost : REAL);
    BEGIN
      MMTHS := 4.5;
      MMHRS := MMTHS/(0.0136*0.033);
    END METHOD;

  ASK METHOD GetReworkRewingTimes(IN IDT : REAL; OUT MMTHS, MMHRS, Cost : REAL);
    BEGIN
      ASK SELF TO GetReworkTimes(IDT,MMTHS,MMHRS,Cost);
      MMTHS := (MMHRS * 0.0136 * 0.033) + 1.0;
    END METHOD;
  ASK METHOD GetReworkRewingUpgradeTimes(IN IDT : REAL;
                                                                        OUT MMTHS, MMHRS, Cost : REAL);
    BEGIN
      ASK SELF TO GetReworkTimes(IDT,MMTHS,MMHRS,Cost);
      MMTHS := (MMHRS * 0.0136 * 0.033) + 1.5;
    END METHOD;
END OBJECT;

END MODULE.
```

```
DEFINITION MODULE Stats;

FROM Aircraft IMPORT AircraftObj;

TYPE
  StatsObj = OBJECT
    SumTAT,SumMMTHS,SumMMHRS,
    SumOfSquaresTAT,SumOfSquaresMMTHS,SumOfSquaresMMHRS,
    MinTAT,MinMMTHS,MinMMHRS,
    MaxTAT,MaxMMTHS,MaxMMHRS: REAL;
    N: INTEGER;
    ASK METHOD ObjInit;
    ASK METHOD GetSample(IN TAT,MMTHS,MMHRS : REAL);
    ASK METHOD MeanTAT() : REAL ;
    ASK METHOD MeanMMTHS() : REAL ;
    ASK METHOD MeanMMHRS() : REAL ;
    ASK METHOD MeanMMHRSYr(IN TotalYears : REAL) : REAL;
    ASK METHOD VarianceTAT() : REAL ;
    ASK METHOD VarianceMMTHS() : REAL ;
    ASK METHOD VarianceMMHRS() : REAL ;
    ASK METHOD StdDevTAT() : REAL ;
    ASK METHOD StdDevMMTHS() : REAL ;
    ASK METHOD StdDevMMHRS() : REAL ;
    ASK METHOD Reset;
  END OBJECT;

  YearlyStatsObj = OBJECT
    SumMMHRS, SumOfSquaresMMHRS,
    SumOfSquaresVisits,
    YearMeanMMHRS, YearStdDevMMHRS,
    YearMeanVisits, YearStdDevVisits  :  REAL;
    N, NYear :  INTEGER;
    ASK METHOD ObjInit;
    ASK METHOD GetSample(IN MMHRS : REAL);
    ASK METHOD GetYearlySample;
    ASK METHOD GetYearlyMeanMMHRS() : REAL;
    ASK METHOD GetYearlyStdDevMMHRS() : REAL;
    ASK METHOD GetYearlyMeanVisits() : REAL;
    ASK METHOD GetYearlyStdDevVisits() : REAL;
    ASK METHOD Reset1;
    ASK METHOD Reset2;
  END OBJECT;

  CumStatsObj = OBJECT
    SumCumTime, SumVisits,
    SumOfSquaresCumTime, SumOfSquaresVisits  :  REAL;
    N  :  INTEGER;
    ASK METHOD ObjInit;
    ASK METHOD GetCumSample(IN Prowler : AircraftObj);
    ASK METHOD MeanCumTime() : REAL;
    ASK METHOD MeanVisits() : REAL;
    ASK METHOD VarianceCumTime() : REAL;
    ASK METHOD VarianceVisits() : REAL;
    ASK METHOD StdDevCumTime() : REAL;
    ASK METHOD StdDevVisits() : REAL;
    ASK METHOD Reset;
  END OBJECT;

END MODULE.
```

```
IMPLEMENTATION MODULE Stats ;

FROM Aircraft IMPORT AircraftObj;
FROM MathMod IMPORT SQRT;
FROM SimMod IMPORT SimTime;

OBJECT StatsObj;
  ASK METHOD ObjInit;
    BEGIN
      Reset;
    END METHOD;

  ASK METHOD GetSample(IN TAT,MMTHS,MMHRS : REAL);
    BEGIN
      SumTAT := SumTAT + TAT;
      SumMMTHS := SumMMTHS + MMTHS;
      SumMMHRS := SumMMHRS + MMHRS;
      SumOfSquaresTAT := SumOfSquaresTAT + TAT*TAT;
      SumOfSquaresMMTHS := SumOfSquaresMMTHS + MMTHS*MMTHS;
      SumOfSquaresMMHRS := SumOfSquaresMMHRS + MMHRS*MMHRS;
      MinTAT := MINOF(TAT,MinTAT);
      MaxTAT := MAXOF(TAT,MaxTAT);
      MinMMTHS := MINOF(MMTHS,MinMMTHS);
      MaxMMTHS := MAXOF(MMTHS,MaxMMTHS);
      MinMMHRS := MINOF(MMHRS,MinMMHRS);
      MaxMMHRS := MAXOF(MMHRS,MaxMMHRS);
      N := N +1;
    END METHOD;

  ASK METHOD MeanTAT() : REAL ;
    BEGIN
      RETURN SumTAT/FLOAT(N);
    END METHOD;

  ASK METHOD MeanMMTHS() : REAL ;
    BEGIN
      RETURN SumMMTHS/FLOAT(N);
    END METHOD;

  ASK METHOD MeanMMHRS() : REAL ;
    BEGIN
      RETURN SumMMHRS/FLOAT(N);
    END METHOD;

  ASK METHOD MeanMMHRSYr(IN TotalYears : REAL) : REAL;
    BEGIN
      RETURN (SumMMHRS/TotalYears);
    END METHOD;

  ASK METHOD VarianceTAT() : REAL ;
    BEGIN
      RETURN (SumOfSquaresTAT - SumTAT*SumTAT/FLOAT(N))/FLOAT(N-1);
    END METHOD;
  ASK METHOD VarianceMMTHS() : REAL ;
    BEGIN
      RETURN (SumOfSquaresMMTHS - SumMMTHS*SumMMTHS/FLOAT(N))/FLOAT(N-1);
    END METHOD;

  ASK METHOD VarianceMMHRS() : REAL ;
    BEGIN
      RETURN (SumOfSquaresMMHRS - SumMMHRS*SumMMHRS/FLOAT(N))/FLOAT(N-1);
    END METHOD;

  ASK METHOD StdDevTAT() : REAL ;
    BEGIN
      RETURN SQRT( VarianceTAT() );
    END METHOD;

  ASK METHOD StdDevMMTHS() : REAL ;
    BEGIN
      RETURN SQRT( VarianceMMTHS() );
    END METHOD;

  ASK METHOD StdDevMMHRS() : REAL ;
    BEGIN
      RETURN SQRT( VarianceMMHRS() );
    END METHOD;

  ASK METHOD Reset;
    BEGIN
      SumTAT := 0.0;
      SumMMTHS := 0.0;
      SumMMHRS := 0.0;
      SumOfSquaresTAT := 0.0;
      SumOfSquaresMMTHS := 0.0;
      SumOfSquaresMMHRS := 0.0;
      MinTAT := MAX(REAL);
      MinMMTHS := MAX(REAL);
      MinMMHRS := MAX(REAL);
      MaxTAT := MIN(REAL);
      MaxMMTHS := MIN(REAL);
      MaxMMHRS := MIN(REAL);
      N := 0;
    END METHOD;
  END OBJECT;

OBJECT  YearlyStatsObj;
  ASK METHOD ObjInit;
    BEGIN
      Reset1;
    END METHOD;

  ASK METHOD GetSample(IN MMHRS : REAL);
    BEGIN
      SumMMHRS := SumMMHRS + MMHRS;
      SumOfSquaresMMHRS := SumOfSquaresMMHRS + MMHRS*MMHRS;
      N := N + 1;
      SumOfSquaresVisits := SumOfSquaresVisits + FLOAT(N*N);
    END METHOD;

  ASK METHOD GetYearlySample;
    BEGIN
      IF SumMMHRS = 0.0
        YearMeanMMHRS := YearMeanMMHRS + 0.0;
        YearStdDevMMHRS := YearStdDevMMHRS + 0.0;
        YearMeanVisits := YearMeanVisits + 0.0;
      ELSE
        YearMeanMMHRS := YearMeanMMHRS + SumMMHRS/FLOAT(N);
        YearMeanVisits := YearMeanVisits + FLOAT(N);
        IF N <= 1
          YearStdDevMMHRS := YearStdDevMMHRS + 0.0;
          YearStdDevVisits := YearStdDevVisits + 0.0;
        ELSE
          YearStdDevMMHRS := YearStdDevMMHRS +
                     SQRT((SumOfSquaresMMHRS - SumMMHRS*SumMMHRS/FLOAT(N))/FLOAT(N-
  1));
          YearStdDevVisits := YearStdDevVisits +
                     SQRT((SumOfSquaresVisits - FLOAT(N*N)/FLOAT(N))/FLOAT(N-1));
        END IF;
      END IF;
      NYear := NYear + 1;
      ASK SELF TO Reset2;
    END METHOD;

  ASK METHOD GetYearlyMeanMMHRS() : REAL;
    BEGIN
      RETURN YearMeanMMHRS/FLOAT(NYear);
    END METHOD;

  ASK METHOD GetYearlyStdDevMMHRS() : REAL;
    BEGIN
      RETURN YearStdDevMMHRS/FLOAT(NYear);
    END METHOD;

  ASK METHOD GetYearlyMeanVisits() : REAL;
    BEGIN
      RETURN YearMeanVisits/FLOAT(NYear);
    END METHOD;

  ASK METHOD GetYearlyStdDevVisits() : REAL;
    BEGIN
      RETURN YearStdDevVisits/FLOAT(NYear);
    END METHOD;

  ASK METHOD Reset1;
    BEGIN
      SumMMHRS := 0.0;
      SumOfSquaresMMHRS := 0.0;
      SumOfSquaresVisits := 0.0;
      YearMeanMMHRS := 0.0;
      YearStdDevMMHRS := 0.0;
```

```
      YearMeanVisits := 0.0;
      YearStdDevVisits := 0.0;
      N := 0;
    END METHOD;

  ASK METHOD Reset2;
    BEGIN
      SumMMHRS := 0.0;
      SumOfSquaresMMHRS := 0.0;
      SumOfSquaresVisits := 0.0;
      N := 0;
    END METHOD;
  END OBJECT;

OBJECT CumStatsObj;
  ASK METHOD ObjInit;
    BEGIN
      Reset;
    END METHOD;

  ASK METHOD GetCumSample(IN Prowler : AircraftObj);
    BEGIN
      SumCumTime := SumCumTime + ASK Prowler TO GetCumTimeInDepot;
      SumVisits := SumVisits + ASK Prowler TO GetNumVisits;
      SumOfSquaresCumTime := SumOfSquaresCumTime + (ASK Prowler TO GetCumTimeInDepot)
                                   *(ASK Prowler TO GetCumTimeInDepot);
      SumOfSquaresVisits := SumOfSquaresVisits + (ASK Prowler TO GetNumVisits)
                                     *(ASK Prowler TO GetNumVisits);
      N := N + 1;
    END METHOD;

  ASK METHOD MeanCumTime() : REAL ;
    BEGIN
      RETURN SumCumTime/FLOAT(N);
    END METHOD;

  ASK METHOD MeanVisits() : REAL ;
    BEGIN
      RETURN SumVisits/FLOAT(N);
    END METHOD;

  ASK METHOD VarianceCumTime() : REAL ;
    BEGIN
      RETURN (SumOfSquaresCumTime - SumCumTime*SumCumTime/FLOAT(N))/FLOAT(N-1);
    END METHOD;

  ASK METHOD VarianceVisits() : REAL ;
    BEGIN
      RETURN (SumOfSquaresVisits - SumVisits*SumVisits/FLOAT(N))/FLOAT(N-1);
    END METHOD;

  ASK METHOD StdDevCumTime() : REAL ;
    BEGIN
      RETURN SQRT( VarianceCumTime() );
    END METHOD;

  ASK METHOD StdDevVisits() : REAL ;
    BEGIN
      RETURN SQRT( VarianceVisits() );
    END METHOD;

  ASK METHOD Reset;
    BEGIN
      SumCumTime := 0.0;
      SumVisits := 0.0;
      SumOfSquaresCumTime := 0.0;
      SumOfSquaresVisits := 0.0;
      N := 0;
    END METHOD;
  END OBJECT;

END MODULE.
```

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center 8725 John J. Kingman Rd., STE 0944 Ft. Belvoir, VA 22060-6218	2
2.	Dudley Knox Library, Naval Postgraduate School 411 Dyer Rd. Monterey, CA 93943-5101	2
3.	Defense Logistics Studies Information Exchange U.S. Army Logistics Management Center Fort Lee, VA 23801	1
4.	Professor Arnold H. Buss, Code OR/Bu Department of Operations Research Naval Postgraduate School Monterey, CA 93943-5000	1
5.	RADM Donald Eaton USN (Ret), Code SM/ET Department of Systems Management Naval Postgraduate School Monterey, CA 93940	1
6.	Professor Samuel H. Parry, Code OR/Py Department of Operations Research Naval Postgraduate School Monterey, CA 93943-5000	1
7.	Mr. Richard Massaro, Air 3.1.1H Naval Air Systems Command Headquarters 1421 Jefferson Davis Highway Arlington, VA 22243	1
8.	LCDR Mark Nye EA-6B Program Office, 04B NADEP Jacksonille, FL 32212-0016	1
9.	Professor David A. Schrady, Code OR/So Department of Operations Research Naval Postgraduate School Monterey, CA 93943-5000	1

10. Professor Lyn R. Whitaker, Code OR/Wh 1
Department of Operations Research
Naval Postgraduate School
Monterey, CA 93943-5000

11. Deputy Chief of Naval Operations (Logistics) 1
ATTN: CDR Robert Drash N422C
Navy Pentagon
Washington, D.C. 20350-2000

12. LCDR Michael D. Walls, USN 1
5708 Caddoan Road
Virginia Beach, VA 23462